```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Media;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using System.Threading;
using System.Timers;

namespace Design_Project
{
    public partial class Form1 : Form
    {
        private MLApp.MLAppClass ml;
        //Used for MATLAB communication.

        static int secDuration = 60;
        static int minDuration = 9;
        int secTime = secDuration;
        // leave seconds timer equal to 60. Counts down from 60 to 0.
        int minTime = minDuration;
        // Total number of minutes to count down **MINUS ONE**.

        static int max = 20;
        //Variable used for testing.
        int testCounter = 0;
        int maxTemp = 0;
        //Variable used in event total number of words is LESS than 20. Used in updating ↙
    labels, etc.

        string testString = "";
        string guessString = "";
        private int[] indexString = new int[4];
        //Initialize indexString to all incorrectly guessed.
        int engWord = 0;
        int checkToggle = 1;

        // Variables used for testing mode.
        // testString/guessString/indexString are used to track correctly guessed letters↙
    .
        // engWord is used to track display type of English letters.
        // engWord = 0 will not show any English letters.
        // engWord = 1 will show the entire word in English.
        // engWord = 2 will only show correctly guessed letters.

        // For minute countdown: If timer starts at 5,
        // takes one minute before changing to 4.
        // So if timer is  desired to be 5:00,
        // start minute timer at 4minutes.

        int[] statsCurrTest = new int[4];
        int[] statsCurrRand = new int[4];
        //Variables used to track current sessions. Will be used to write into a file    ↙
    later.

        int[] statsArrTest = new int[4];
        int[] statsArrRandom = new int[4];
        // Variables used for keeping track of stats.

        int[] arr = new int[150];

        SoundPlayer soundByte = null;
    int navButton = 0;
    //Used to track/enable navigation of specific buttons.

    private void Form1_Load(object sender, EventArgs e)
     {
         // all of the options for a serial device
         // can be sent through the constructor of the SerialPort class
         // PortName = "COM1", Baud Rate = 19200, Parity = None,
         // Data Bits = 8, Stop Bits = One, Handshake = None

         serialPort1.DataReceived += new SerialDataReceivedEventHandler        ↙
(sp_DataReceived);
         serialPort1.PortName = "COM4";
     }

    void sp_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs  ↙
e)
     {//Method used to handle incoming button presses from microcontroller.
        //Microcontroller is programmed to send serial data upon press.

         byte[] test = new byte[1];
         char var = Convert.ToChar(serialPort1.ReadChar());

         if (var == 'a')
         {
             if (navButton == 1)
             {
                 BeginInvoke(new MethodInvoker(delegate
                 {//Needed for accessing objects when taking in serial data.
                     button15_Click(null, null);
                 }));
             }
             else
             {// Treats button as a method of entering SpeechRec
                 BeginInvoke(new MethodInvoker(delegate
                 {//Needed for accessing objects when taking in serial data.
                     if (listBoxTest.Items.Count != 0)
                     {// Check if perosn is currently in test mode
                         statsCurrRand[0] = 0;
                         statsCurrRand[1] = 0;
                         statsCurrRand[2] = 0;
                         statsCurrRand[3] = 0;
                         statsRandUpdate();

                         statsCurrTest[0] = 0;
                         statsCurrTest[1] = 0;
                         statsCurrTest[2] = 0;
                         statsCurrTest[3] = 0;
                         statsTestUpdate();

                         timerReset();
                         panelHideAll();
                     }
                     panelSpeech();
                     navButton = 1;
                 }));
             }
         }
         else if (var == 'b')
         {
             if (navButton == 1)
             {//Treats button as if moving to "Next Word"
                 BeginInvoke(new MethodInvoker(delegate
                 {//Needed for accessing objects when taking in serial data.
                     button9_Click(null,null);
                 }));
             }
         }
        else if (var == 'c')
        {
            if (navButton == 1)
            {//Treats button as if recording Speech
                BeginInvoke(new MethodInvoker(delegate
                {//Needed for accessing objects when taking in serial data.
                    button47_Click(null, null);
                }));
            }
        }
        else if (var == 'd')
        {
            if (navButton == 1)
            {//Treats button as if Verifying answer
                BeginInvoke(new MethodInvoker(delegate
                {//Needed for accessing objects when taking in serial data.
                    speechVerify();
                }));
            }
        }
    }



    public Form1()
    {
        InitializeComponent();
        try
        {
            serialPort1.Open();
            // open port
        }
        catch (Exception )
        {
            // handle the exception
        }

        panelNav.Show();
        panelNav.Dock = DockStyle.Fill;
        //Brings main navigation GUI to front.
    }

    //**Methods for statistics tracking start here
    private void statsFileRead()
    {//Used to read in stats for both Test and Random modes.

        double frac = 0;
        string temp = "";

        //Following code reads in Test Stats.
        FileInfo file = new FileInfo("Stats\\statsTest.txt");
        StreamReader statRead = file.OpenText();
        for (int i = 0; i < 4; i++)
        {
            temp = Convert.ToString(statRead.ReadLine());
            statsArrTest[i] = Convert.ToInt32(temp);
        }


        label26.Text = statsArrTest[0].ToString();
        label24.Text = statsArrTest[1].ToString();
        label14.Text = statsArrTest[2].ToString();
        label82.Text = statsArrTest[3].ToString();

        if (statsArrTest[2] != 0)
        {//Checks for situation where stats are nonexistant or reset.
         //Following code assumes statistics are non-zero.
```

```
            frac = Convert.ToDouble(statsArrTest[0]) / Convert.ToDouble(statsArrTest ↙
[2]);
            frac = frac * 100;
            label32.Text = string.Format("{0:0.00}", frac);
            frac = Convert.ToDouble(statsArrTest[1]) / Convert.ToDouble(statsArrTest ↙
[2]);
            frac = frac * 100;
            label31.Text = string.Format("{0:0.00}", frac);
            frac = Convert.ToDouble(statsArrTest[3]) / Convert.ToDouble(statsArrTest ↙
[2]);
            frac = frac * 100;
            label81.Text = string.Format("{0:0.00}", frac);
        }
        else
        {//Prevents displaying text; dividing by zero.

            label81.Text = "--";
            label32.Text = "--";
            label31.Text = "--";
        }
        statRead.Close();

        //Following code reads in Random stats.
        FileInfo file2 = new FileInfo("Stats\\statsRand.txt");
        StreamReader statRead2 = file2.OpenText();
        for (int i = 0; i < 4; i++)
        {
            temp = Convert.ToString(statRead2.ReadLine());
            statsArrRandom[i] = Convert.ToInt32(temp);
        }


        label53.Text = statsArrRandom[0].ToString();
        label51.Text = statsArrRandom[1].ToString();
        label49.Text = statsArrRandom[2].ToString();
        label90.Text = statsArrRandom[3].ToString();

        if (statsArrRandom[2] != 0)
        {//Checks for situation where stats are nonexistant or reset.
         //Following code assumes statistics are non-zero.

            frac = Convert.ToDouble(statsArrRandom[0]) / Convert.ToDouble             ↙
(statsArrRandom[2]);
            frac = frac * 100;
            label45.Text = string.Format("{0:0.00}", frac);
            frac = Convert.ToDouble(statsArrRandom[1]) / Convert.ToDouble             ↙
(statsArrRandom[2]);
            frac = frac * 100;
            label33.Text = string.Format("{0:0.00}", frac);
            frac = Convert.ToDouble(statsArrRandom[3]) / Convert.ToDouble             ↙
(statsArrRandom[2]);
            frac = frac * 100;
            label89.Text = string.Format("{0:0.00}", frac);
        }
        else
        {//Prevents displaying text; dividing by zero.

            label45.Text = "--";
            label33.Text = "--";
            label89.Text = "--";
        }
        statRead2.Close();

        //Following code writes in current session details.
        label7.Text = "0";
        label78.Text = "0";
        label10.Text = "0";
        label12.Text = "0";
```

```
            label29.Text = "--";
            label30.Text = "--";
            label77.Text = "--";

            label86.Text = "0";
            label59.Text = "0";
            label57.Text = "0";
            label55.Text = "0";
            label47.Text = "--";
            label46.Text = "--";
            label85.Text = "--";
        }

        private void statsTestRec()
        {//Records stats of Timed Test to File.
            StreamWriter statRead; // Creates streamWriter to write to the necessary file↙
.
            statRead = File.CreateText("Stats\\statsTest.txt");
            statsArrTest[0] += statsCurrTest[0]; // Current Correct
            statsArrTest[1] += statsCurrTest[1]; // Current skipped
            statsArrTest[2] += statsCurrTest[2]; // Current total words attempted
            statsArrTest[3] += statsCurrTest[3]; // Current incorrect

            //Clears current statistics
            statsCurrTest[0] = 0;
            statsCurrTest[1] = 0;
            statsCurrTest[2] = 0;
            statsCurrTest[3] = 0;

            //Writes statistics to file.
            string Temp = statsArrTest[0].ToString();
            for (int i = 1; i < 4; i++)
            {
                Temp = Temp + System.Environment.NewLine + statsArrTest[i].ToString();
            }

            //Dumps out streamWriter and closes.
            statRead.Write(Temp);
            statRead.Flush();
            statRead.Close();
        }

        private void statsTestUpdate()
        {//Method used to update current stats as a test occurs.

            if (statsCurrTest[2] != 0)
            {
                double frac = 0;

                //Takes counters and adds to statistics strings.
                label7.Text = statsCurrTest[0].ToString();
                label10.Text = statsCurrTest[1].ToString();
                label12.Text = statsCurrTest[2].ToString();
                label78.Text = statsCurrTest[3].ToString();

                //Updates the display for fractions, to 2 decimal places.
                frac = Convert.ToDouble(statsCurrTest[0]) / Convert.ToDouble            ↙
(statsCurrTest[2]);
                frac = frac * 100;
                label30.Text = string.Format("{0:0.00}", frac);
                frac = Convert.ToDouble(statsCurrTest[1]) / Convert.ToDouble            ↙
(statsCurrTest[2]);
                frac = frac * 100;
                label29.Text = string.Format("{0:0.00}", frac);
                frac = Convert.ToDouble(statsCurrTest[3]) / Convert.ToDouble            ↙
(statsCurrTest[2]);
                frac = frac * 100;
                label77.Text = string.Format("{0:0.00}", frac);
        }
        else
        {
            label7.Text = "0";
            label78.Text = "0";
            label10.Text = "0";
            label12.Text = "0";
            label29.Text = "--";
            label30.Text = "--";
            label77.Text = "--";
        }

    }

    private void statsRandRec()
    {//Method used to records stats of Timed Test to File.
        StreamWriter statRead; // Creates streamWriter to write to the necessary file↙
.
        statRead = File.CreateText("Stats\\statsRand.txt");

        //Updates all-time statistics for correct/skipped/total words.
        statsArrRandom[0] += statsCurrRand[0];
        statsArrRandom[1] += statsCurrRand[1];
        statsArrRandom[2] += statsCurrRand[2];
        statsArrRandom[3] += statsCurrRand[3];

        //Resets current session statistics.
        statsCurrRand[0] = 0;
        statsCurrRand[1] = 0;
        statsCurrRand[2] = 0;
        statsCurrRand[3] = 0;

        //Writes statistics to file.
        string Temp = statsArrRandom[0].ToString();
        for (int i = 1; i < 4; i++)
        {
            Temp = Temp + System.Environment.NewLine + statsArrRandom[i].ToString();
        }

        //Dumps streamWriter and closes it.
        statRead.Write(Temp);
        statRead.Flush();
        statRead.Close();

    }

    private void statsRandUpdate()
    {//Method used to update current stats as a random test occurs.
        if (statsCurrRand[2] != 0)
        {
            double frac = 0;

            //Takes counters and adds to statistics strings.
            label59.Text = statsCurrRand[0].ToString();
            label57.Text = statsCurrRand[1].ToString();
            label55.Text = statsCurrRand[2].ToString();
            label86.Text = statsCurrRand[3].ToString();

            //Updates the display for fractions, to 2 decimal places.
            frac = Convert.ToDouble(statsCurrRand[0]) / Convert.ToDouble          ↙
(statsCurrRand[2]);
            frac = frac * 100;
            label47.Text = string.Format("{0:0.00}", frac);
            frac = Convert.ToDouble(statsCurrRand[1]) / Convert.ToDouble          ↙
(statsCurrRand[2]);
            frac = frac * 100;
            label46.Text = string.Format("{0:0.00}", frac);
            frac = Convert.ToDouble(statsCurrRand[3]) / Convert.ToDouble          ↙
```

```
(statsCurrRand[2]);
            frac = frac * 100;
            label85.Text = string.Format("{0:0.00}", frac);
        }
        else
        {
            label86.Text = "0";
            label59.Text = "0";
            label57.Text = "0";
            label55.Text = "0";
            label47.Text = "--";
            label46.Text = "--";
            label85.Text = "--";
        }

    }
    //Methods for statistics tracking end here

    //**Methods for adding to Lists start here
    public void allAdd(int length)
    {//Method used to add a word, when all 1/2/3/4 letter words are in dictionary.
     //Does not include capital letters, puncutation etc.

        // Creates an empty file; nothing is to be written in it.
        StreamWriter stRead;
        stRead = File.CreateText("Dictionary\\empty1.txt");

        string Temp = "";
        int listCount = 0;
        int wordTotal = 0;

        if (length == 1) // Checks length to write to proper file
        {// Identifies word length = 1. Adds word to listbox and updates the total   ↙
word count.

            stRead = File.CreateText("Dictionary\\one.txt");
            listBoxONE.Items.Add(textBox3.Text);
            wordTotal = Convert.ToInt32(listBoxONE.Items.Count);
        }
        else if (length == 2)
        {// Identifies word length = 2. Adds word to listbox and updates the total   ↙
word count.

            stRead = File.CreateText("Dictionary\\two.txt");
            listBoxTWO.Items.Add(textBox3.Text);
            wordTotal = Convert.ToInt32(listBoxTWO.Items.Count);
        }
        else if (length == 3)
        {// Identifies word length = 3. Adds word to listbox and updates the total   ↙
word count.

            stRead = File.CreateText("Dictionary\\three.txt");
            listBoxTHREE.Items.Add(textBox3.Text);
            wordTotal = Convert.ToInt32(listBoxTHREE.Items.Count);
        }
        else if (length == 4)
        {// Identifies word length = 4. Adds word to listbox and updates the total   ↙
word count.

            stRead = File.CreateText("Dictionary\\four.txt");
            listBoxFOUR.Items.Add(textBox3.Text);
            wordTotal = Convert.ToInt32(listBoxFOUR.Items.Count);
        }

        for (listCount = 0; listCount <= wordTotal - 1; listCount++)
        { // For loop to update the respective list of words.

            if ((listCount != 0) && length == 1)
            { // For every loop from listCount == 1 onward, adds each word to the    ↙
file.

                Temp = Temp + System.Environment.NewLine + listBoxONE.Items          ↙
[listCount].ToString();
            }
            else if ((listCount != 0) && length == 2)
            {// For every loop from listCount == 2 onward, adds each word to the file↙
.

                Temp = Temp + System.Environment.NewLine + listBoxTWO.Items          ↙
[listCount].ToString();
            }
            else if ((listCount != 0) && length == 3)
            {// For every loop from listCount == 3 onward, adds each word to the file↙
.

                Temp = Temp + System.Environment.NewLine + listBoxTHREE.Items        ↙
[listCount].ToString();
            }
            else if ((listCount != 0) && length == 4)
            {// For every loop from listCount == 4 onward, adds each word to the file↙
.

                Temp = Temp + System.Environment.NewLine + listBoxFOUR.Items         ↙
[listCount].ToString();
            }
            else
            { // Else statement used to write the initial line to the file.
              // Dependant on initial length passed to method.

                if (length == 1)
                {
                    Temp = listBoxONE.Items[0].ToString();
                }
                else if (length == 2)
                {
                    Temp = listBoxTWO.Items[0].ToString();
                }
                else if (length == 3)
                {
                    Temp = listBoxTHREE.Items[0].ToString();
                }
                else if (length == 4)
                {
                    Temp = listBoxFOUR.Items[0].ToString();
                }
            }

        }

        // Updates respective files as necessary and closes stRead.
        stRead.Write(Temp);
        stRead.Flush();
        stRead.Close();
        textBox3.Clear();
    }

    public void oneAdd(int length)
    {//Method used to add a word, when only 1 letter words are in dictionary.
     //Does not include capital letters, puncutation etc.

        // Creates streamWriter to write to the necessary file.
        StreamWriter stRead;
        stRead = File.CreateText("Dictionary\\one.txt");
        string Temp = "";
        int wordTotal = Convert.ToInt32(listBox2.Items.Count);
        int listCount = 0;
        for (listCount = 0; listCount <= wordTotal - 1; listCount++)
        {// For loop to write each existing word back into the file, along with the  ↙
new word.
            if (listCount != 0)
            { // Used to write every word that appears after the first word.
                Temp = Temp + System.Environment.NewLine + listBox2.Items[listCount].↙
ToString();
            }
            else
            { // Used to write the first word that appears at the top of the list.
              // Note that the listbox is to be sorted alphabetically.
                Temp = listBox2.Items[0].ToString();
            }
        }

        // Updates respective files as necessary and closes stRead.
        stRead.Write(Temp);
        stRead.Flush();
        stRead.Close();
        textBox3.Clear();
    }

    public void twoAdd(int length)
    {//Method used to add a word, when only 2 letter words are in dictionary.
     //Does not include capital letters, puncutation etc.

        // Creates streamWriter to write to the necessary file.
        StreamWriter stRead;
        stRead = File.CreateText("Dictionary\\two.txt");
        string Temp = "";
        int wordTotal = Convert.ToInt32(listBox2.Items.Count);
        int listCount = 0;

        for (listCount = 0; listCount <= wordTotal - 1; listCount++)
        {// For loop to write each existing word back into the file, along with the  ↙
new word.
            if (listCount != 0)
            {// Used to write every word that appears after the first word.
                Temp = Temp + System.Environment.NewLine + listBox2.Items[listCount].↙
ToString();
            }
            else
            { // Used to write the first word that appears at the top of the list.
              // Note that the listbox is to be sorted alphabetically.
                Temp = listBox2.Items[0].ToString();
            }
        }

        // Updates respective files as necessary and closes stRead.
        stRead.Write(Temp);
        stRead.Flush();
        stRead.Close();
        textBox3.Clear();
    }

    public void threeAdd(int length)
    { //Method used to add a word, when only 3 letter words are in dictionary.
       //Does not include capital letters, puncutation etc.

        // Creates streamWriter to write to the necessary file.
        StreamWriter stRead;
        stRead = File.CreateText("Dictionary\\three.txt");
        string Temp = "";
        int wordTotal = Convert.ToInt32(listBox2.Items.Count);
        int listCount = 0;

        for (listCount = 0; listCount <= wordTotal - 1; listCount++)
    {// For loop to write each existing word back into the file, along with the  ↙
new word.
      if (listCount != 0)
      {// Used to write every word that appears after the first word.
        Temp = Temp + System.Environment.NewLine + listBox2.Items[listCount].↙
ToString();
      }
      else
      { // Used to write the first word that appears at the top of the list.
        // Note that the listbox is to be sorted alphabetically.
        Temp = listBox2.Items[0].ToString();
      }
    }

    // Updates respective files as necessary and closes stRead.
    stRead.Write(Temp);
    stRead.Flush();
    stRead.Close();
    textBox3.Clear();
  }

  public void fourAdd(int length)
  { //Method used to add a word, when only 4 letter words are in dictionary.
   //Does not include capital letters, puncutation etc.

    // Creates streamWriter to write to the necessary file.
    StreamWriter stRead;
    stRead = File.CreateText("Dictionary\\four.txt");
    string Temp = "";
    int wordTotal = Convert.ToInt32(listBox2.Items.Count);
    int listCount = 0;

    for (listCount = 0; listCount <= wordTotal - 1; listCount++)
    {// For loop to write each existing word back into the file, along with the  ↙
new word.
      if (listCount != 0)
      {// Used to write every word that appears after the first word.
        Temp = Temp + System.Environment.NewLine + listBox2.Items[listCount].↙
ToString();
      }
      else
      { // Used to write the first word that appears at the top of the list.
        // Note that the listbox is to be sorted alphabetically.
        Temp = listBox2.Items[0].ToString();
      }
    }

    // Updates respective files as necessary and closes stRead.
    stRead.Write(Temp);
    stRead.Flush();
    stRead.Close();
    textBox3.Clear();
  }

  public void numAdd(int length)
  {   //Method used to add a word, when only numbers are in dictionary.

    // Creates streamWriter to write to the necessary file.
    StreamWriter stRead;
    stRead = File.CreateText("Dictionary\\numbers.txt");
    string Temp = "";
    int wordTotal = Convert.ToInt32(listBox2.Items.Count);
    int listCount = 0;

    for (listCount = 0; listCount <= wordTotal - 1; listCount++)
    {// For loop to write each existing number back into the file, along with the↙
 new number.
      if (listCount != 0)
```

```
            {// Used to write every number that appears after the first number.
                Temp = Temp + System.Environment.NewLine + listBox2.Items[listCount].↙
ToString();
            }
            else
            {   // Used to write the first number that appears at the top of the list↙
.
                // Note that the listbox is to be sorted alphabetically, **NOT       ↙
NUMERICALLY*.
                Temp = listBox2.Items[0].ToString();
            }
        }

        // Updates respective files as necessary and closes stRead.
        stRead.Write(Temp);
        stRead.Flush();
        stRead.Close();
        textBox3.Clear();
    }

    public void capAdd(int length)
    { //Method used to add a word, when only word containing capital letters are in  ↙
dictionary.

        // Creates streamWriter to write to the necessary file.
        StreamWriter stRead;
        stRead = File.CreateText("Dictionary\\capitals.txt");
        string Temp = "";
        int wordTotal = Convert.ToInt32(listBox2.Items.Count);
        int listCount = 0;

        for (listCount = 0; listCount <= wordTotal - 1; listCount++)
        {// For loop to write each existing word back into the file, along with the  ↙
new word.
            if (listCount != 0)
            {// Used to write every number that appears after the first number.
                Temp = Temp + System.Environment.NewLine + listBox2.Items[listCount].↙
ToString();
            }
            else
            {  // Used to write the first number that appears at the top of the list.
                // Note that the listbox is to be sorted alphabetically.
                Temp = listBox2.Items[0].ToString();
            }
        }

        // Updates respective files as necessary and closes stRead.
        stRead.Write(Temp);
        stRead.Flush();
        stRead.Close();
        textBox3.Clear();
    }

    //checkAddLetter may be useless.
    public void checkAddLetter()
    {
        int listCount = Convert.ToInt32(listBox2.Items.Count);
        int length = textBox3.Text.Length;

        int charCheck = -4;
        charCheck = checkCharacter();

        if (textBoxALLCH.Text == "1" && length < 5 && charCheck == 0)
        {
            listBox2.Items.Add(textBox3.Text);
            allAdd(length);
        }
        else if (length == 1 && charCheck == 0)
        {
            listBox2.Items.Add(textBox3.Text);
            oneAdd(length);
        }
        else if (length == 2 && charCheck == 0)
        {
            listBox2.Items.Add(textBox3.Text);
            twoAdd(length);
        }
        else if (length == 3 && charCheck == 0)
        {
            listBox2.Items.Add(textBox3.Text);
            threeAdd(length);
        }
        else if (length == 4 && charCheck == 0)
        {
            listBox2.Items.Add(textBox3.Text);
            fourAdd(length);
        }
        else {}
    }

    public void checkAdd()
    {// Method used to try and add a word to the dictionary.

        bool mixCheck = false;
        bool numericCheck = false;
        bool capCheck = false;
        int listCount = Convert.ToInt32(listBox2.Items.Count);
        string temp = textBox3.Text;
        int length = textBox3.Text.Length;

        mixCheck = System.Text.RegularExpressions.Regex.IsMatch(textBox3.Text, "^[a- ↙
zA-Z_0-9]");
        numericCheck = System.Text.RegularExpressions.Regex.IsMatch(textBox3.Text, "^↙
[0-9]");
        capCheck = System.Text.RegularExpressions.Regex.IsMatch(textBox3.Text, "^[a- ↙
zA-Z]");
        if (System.Text.RegularExpressions.Regex.IsMatch(temp, "^[0-9]") && !        ↙
mixCheck)
        {// Checks if word to be added is strictly a number.
            //Implies word is strictly numerical.
            if (listBox2.Items.Contains(Convert.ToInt32(temp)))
            {
                MessageBox.Show("Error: Number already exists.");
            }
            else if (textBoxNUMCH.Text == "1")
            {
                listBox2.Items.Add(textBox3.Text);
            }
        }
        else if (System.Text.RegularExpressions.Regex.IsMatch(temp, "^[a-zA-Z]") && !↙
mixCheck)
        {
            listBox2.Items.Add(textBox3.Text);
        }
        else if (numericCheck)
        {
            listBox2.Items.Add(textBox3.Text);
        }
        else if (capCheck)
        {
            listBox2.Items.Add(textBox3.Text);
        }
        else
        {
            MessageBox.Show("Word contains invalid characters!");
            textBox3.Clear();
            return;
        }

        //Checks if the word contains valid characters.
        int charCheck = -4;
        if (!mixCheck)
        {
            charCheck = checkCharacter();
        }
        else if (numericCheck)
        {
            charCheck = checkCharacter();
        }
        else if (capCheck)
        {
            charCheck = checkCharacter();
        }
        else
        {
            textBox3.Clear();
            charCheck = -1;
            length = 3;
        }

        if (charCheck == 1 && textBoxNUMCH.Text == "1" && length < 4 && length > 0 &&↙
 numericCheck)
        { // charCheck == 1 implies that entered string is stricly numerical.
            numericList();
            numAdd(length);
        }
        else if (textBoxALLCH.Text == "1" && length < 5 && length > 0 && !            ↙
numericCheck && !mixCheck)
        { // Checks the length and desired list to add the word to.
            allAdd(length);
        }
        else if (textBoxONECH.Text == "1" && length == 1 && charCheck != 2 && !       ↙
numericCheck && !mixCheck)
        {
            oneAdd(length);
        }
        else if (textBoxTWOCH.Text == "1" && length == 2 && charCheck != 2 && !       ↙
numericCheck && !mixCheck)
        {
            twoAdd(length);
        }
        else if (textBoxTHREECH.Text == "1" && length == 3 && charCheck != 2 && !     ↙
numericCheck && !mixCheck)
        {
            threeAdd(length);
        }
        else if (textBoxFOURCH.Text == "1" && length == 4 && charCheck != 2 && !      ↙
numericCheck && !mixCheck)
        {
            fourAdd(length);
        }
        else if (textBoxCAPCH.Text == "1" && length > 0 && length < 4 && charCheck ==↙
 2 && capCheck)
        {
            capAdd(length);
        }
        else if (charCheck == -1)
        { // charCheck == -1 implies that entered string contains invalid characters
            // specifically, anything that is not a lower case letter or a number.
            listBox2.Items.Remove(textBox3.Text);
            textBox3.Clear();
            MessageBox.Show("Error: Cannot add word. Word contains invalid characters↙
!");
        }
        else if (charCheck == -3)
        {// charCheck == -3 implies that a number 4 digits or longer wants to be  ↙
added.
            // Program can only handle three digits or less.
            listBox2.Items.Remove(textBox3.Text);
            textBox3.Clear();
            MessageBox.Show("Error: Cannot add numbers longer than three digits!");
        }
        else
        {
            if (length == 0)
            { //Implies nothing was entered.
                listBox2.Items.Remove(textBox3.Text);
                MessageBox.Show("Error: Invalid Word. Improper word length!");
            }
            else if (textBoxNUMCH.Text == "1" && !numericCheck)
            {
                listBox2.Items.Remove(textBox3.Text);
                MessageBox.Show("Error: Invalid word. Cannot add letters to the  ↙
numerical list!");
            }
            else if (charCheck == 2)
            { //Implies word containing a capital letter was entered on the wrong  ↙
list.
                listBox2.Items.Remove(textBox3.Text);
                MessageBox.Show("Error: Invalid Word. Cannot contain a capital letter↙
!");
            }
            else if (textBoxONECH.Text == "1" || textBoxTWOCH.Text == "1" ||
                textBoxTHREECH.Text == "1" || textBoxFOURCH.Text == "1" ||
                textBoxALLCH.Text == "1" || (textBoxPUNCH.Text == "1" && length > 3))
            { // Implies word contains invalid length for list it is trying to be  ↙
added to.
                listBox2.Items.Remove(textBox3.Text);
                MessageBox.Show("Error: Invalid Word. Improper word for this list.");
            }
            else if (charCheck == -3)
            {// charCheck == -3 implies that a number 4 digits or longer wants to be ↙
added.
                // Program can only handle three digits or less.
                listBox2.Items.Remove(textBox3.Text);
                MessageBox.Show("Error: Cannot add numbers longer than three digits! ↙
");
            }
            else if (numericCheck && textBoxNUMCH.Text != "1")
            {
                listBox2.Items.Remove(textBox3.Text);
                MessageBox.Show("Error: Cannot add numbers to alphabetical list!");
            }
            else
            {//Prevents adding a word when no dictionary list has been loaded.
                listBox2.Items.Remove(textBox3.Text);
                MessageBox.Show("Error: Cannot add word. No list chosen!");
            }
            textBox3.Clear();
        }

    }

    private void testAdd(int var)
    {//Method used to randomly add words into a list for Timed Testing.
     //Timed Testing is set to a 10min duration.

        //Maximum possible # of words = 75, since it is expected that
        //Most people will only get to 30-50 words in a 10min duration.
        listBoxTest.Items.Clear();
        int countDown = 0;
        int maxWords = 75;
```

```
string temp = "";

//Checks length of words to be added.
if (var == 1)
{//Checks variable passed to the random generator. var == 1
 //implies the alphabet.

    countDown = listBoxONE.Items.Count;
    //No maximum threshold needed, all 26 letters to be added.
    while (countDown > 0)
    {//Randomly adds words to the list.
        Random objRan = new Random();
        int rand = objRan.Next(0, listBoxONE.Items.Count);
        temp = Convert.ToString(listBoxONE.Items[rand]);
        listBoxONE.Items.Remove(temp);
        listBoxTest.Items.Add(temp);
        countDown--;
    }

}
else if (var == 2)
{//Checks variable passed to the random generator. var == 2
 //implies words with 2 letters.

    countDown = listBoxTWO.Items.Count;
    if (countDown > maxWords)
    {
        countDown = maxWords;
        //Maximum of 75 words to be added.
    }
    while (countDown > 0)
    {//Randomly adds words to the list.
        Random objRan = new Random();
        int rand = objRan.Next(0, listBoxTWO.Items.Count);
        temp = Convert.ToString(listBoxTWO.Items[rand]);
        listBoxTWO.Items.Remove(temp);
        listBoxTest.Items.Add(temp);
        countDown--;
    }
}
else if (var == 3)
{//Checks variable passed to the random generator. var == 3
 //implies words with 3 letters.

    countDown = listBoxTHREE.Items.Count;
    if (countDown > maxWords)
    {
        countDown = maxWords;
        //Maximum of 75 words to be added.
    }
    while (countDown > 0)
    {//Randomly adds words to the list.
        Random objRan = new Random();
        int rand = objRan.Next(0, listBoxTHREE.Items.Count);
        temp = Convert.ToString(listBoxTHREE.Items[rand]);
        listBoxTHREE.Items.Remove(temp);
        listBoxTest.Items.Add(temp);
        countDown--;
    }
}
else if (var == 4)
{//Checks variable passed to the random generator. var == 4
 //implies words with 4 letters.

    countDown = listBoxFOUR.Items.Count;
    if (countDown > maxWords)
    {
        countDown = maxWords;
```

```
                //Maximum of 75 words to be added.
            }
            while (countDown > 0)
            {//Randomly adds words to the list.
                Random objRan = new Random();
                int rand = objRan.Next(0, listBoxFOUR.Items.Count);
                temp = Convert.ToString(listBoxFOUR.Items[rand]);
                listBoxFOUR.Items.Remove(temp);
                listBoxTest.Items.Add(temp);
                countDown--;
            }
        }
        else if (var == 5)
        {//Checks variable passed to the random generator. var == 5
         //implies words with 1-4 letters. No numbers, punctuation, etc.

            countDown = listBoxALL.Items.Count;
            if (countDown > maxWords)
            {
                countDown = maxWords;
                //Maximum of 75 words to be added.
            }
            while (countDown > 0)
            {//Randomly adds words to the list.
                Random objRan = new Random();
                int rand = objRan.Next(0, listBoxALL.Items.Count);
                temp = Convert.ToString(listBoxALL.Items[rand]);
                listBoxALL.Items.Remove(temp);
                listBoxTest.Items.Add(temp);
                countDown--;
            }
        }

        else if (var == 6)
        {//Checks variable passed to the random generator. var == 6
         //implies dealing with numbers only.

            countDown = listBoxNUM.Items.Count;
            if (countDown > maxWords)
            {
                countDown = maxWords;
                //Maximum of 75 numbers to be added.
            }
            while (countDown > 0)
            {//Randomly adds numbers to the list.
                Random objRan = new Random();
                int rand = objRan.Next(0, listBoxNUM.Items.Count);
                temp = Convert.ToString(listBoxNUM.Items[rand]);
                listBoxNUM.Items.Remove(temp);
                listBoxTest.Items.Add(temp);
                countDown--;
            }
        }
    }

    private void randomAdd(int var)
    {//Method used to randomly generate a list of words for Random Test mode.
     //Random Test mode has a cutoff of 20 words/numbers.
        maxTemp = 0;

        listBoxTest.Items.Clear();
        int countDown = 0;
        string temp = "";
        if (var == 1)
        {//Checks variable passed to the random generator. var == 1
         //implies the alphabet.

            countDown = listBoxONE.Items.Count;
    if (countDown > max)
    {
        countDown = max;
        //Maximum of 20 alphabet letters to be added.
    }
    maxTemp = countDown;

    while (countDown > 0)
    {//Randomly adds words to the list.
        Random objRan = new Random();
        int rand = objRan.Next(0, listBoxONE.Items.Count);
        temp = Convert.ToString(listBoxONE.Items[rand]);
        listBoxONE.Items.Remove(temp);
        listBoxTest.Items.Add(temp);
        countDown--;
    }

}
else if (var == 2)
{//Checks variable passed to the random generator. var == 2
 //implies words with 2 letters.

    countDown = listBoxTWO.Items.Count;
    if (countDown > max)
    {
        countDown = max;
        //Maximum of 20 words to be added.
    }
    maxTemp = countDown;

    while (countDown > 0)
    {//Randomly adds words to the list.
        Random objRan = new Random();
        int rand = objRan.Next(0, listBoxTWO.Items.Count);
        temp = Convert.ToString(listBoxTWO.Items[rand]);
        listBoxTWO.Items.Remove(temp);
        listBoxTest.Items.Add(temp);
        countDown--;
    }
}

else if (var == 3)
{//Checks variable passed to the random generator. var == 3
 //implies words with 3 letters.

    countDown = listBoxTHREE.Items.Count;
    if (countDown > max)
    {
        countDown = max;
        //Maximum of 20 words to be added.
    }
    maxTemp = countDown;
    while (countDown > 0)
    {//Randomly adds words to the list.
        Random objRan = new Random();
        int rand = objRan.Next(0, listBoxTHREE.Items.Count);
        temp = Convert.ToString(listBoxTHREE.Items[rand]);
        listBoxTHREE.Items.Remove(temp);
        listBoxTest.Items.Add(temp);
        countDown--;
    }
}

else if (var == 4)
{//Checks variable passed to the random generator. var == 4
 //implies words with 4 letters.

    countDown = listBoxFOUR.Items.Count;
```

```
        if (countDown > max)
        {
            countDown = max;
            //Maximum of 20 words to be added.
        }
        maxTemp = countDown;
        while (countDown > 0)
        {//Randomly adds words to the list.
            Random objRan = new Random();
            int rand = objRan.Next(0, listBoxFOUR.Items.Count);
            temp = Convert.ToString(listBoxFOUR.Items[rand]);
            listBoxFOUR.Items.Remove(temp);
            listBoxTest.Items.Add(temp);
            countDown--;
        }
    }

    else if (var == 5)
    {//Checks variable passed to the random generator. var == 5
     //implies words with 1-4 letters. No numbers, punctuation, etc.

        countDown = listBoxALL.Items.Count;
        if (countDown > max)
        {
            countDown = max;
            //Maximum of 20 words to be added.
        }
        maxTemp = countDown;
        while (countDown > 0)
        {//Randomly adds words to the list.
            Random objRan = new Random();
            int rand = objRan.Next(0, listBoxALL.Items.Count);
            temp = Convert.ToString(listBoxALL.Items[rand]);
            listBoxALL.Items.Remove(temp);
            listBoxTest.Items.Add(temp);
            countDown--;
        }
    }
    else if (var == 6)
    {//Checks variable passed to the random generator. var == 6
     //implies dealing with numbers only.
        countDown = listBoxNUM.Items.Count;
        if (countDown > max)
        {
            countDown = max;
            //Maximum of 20 numbers to be added.
        }
        maxTemp = countDown;
        while (countDown > 0)
        {//Randomly adds numbers to the list.
            Random objRan = new Random();
            int rand = objRan.Next(0, listBoxNUM.Items.Count);
            temp = Convert.ToString(listBoxNUM.Items[rand]);
            listBoxNUM.Items.Remove(temp);
            listBoxTest.Items.Add(temp);
            countDown--;
        }
    }

    //Sets the proper upper limit to be displayed on the screen.
    label44.Text = "  / " + Convert.ToString(maxTemp);
}
// Methods for adding to Lists end here

//**Methods for removing from Lists start here
public void allRem(int length)
{//Method used to remove a word, when the loaded dictionary
 //consists of all words 1-4 letters long.
```

```
        StreamWriter stRead;
        stRead = File.CreateText("Dictionary\\empty2.txt");
        // Creates an empty file; nothing is to be written in it.
        string Temp = "";
        int listCount = 0;
        int wordTotal = 0;

        // Checks length and prepares to write to the correct file.
        if (length == 1)
        {
            stRead = File.CreateText("Dictionary\\one.txt");
            listBoxONE.Items.Remove(textBox3.Text);
            wordTotal = Convert.ToInt32(listBoxONE.Items.Count);
            // Removes word to listbox and updates the total word count.
        }
        else if (length == 2)
        {
            stRead = File.CreateText("Dictionary\\two.txt");
            listBoxTWO.Items.Remove(textBox3.Text);
            wordTotal = Convert.ToInt32(listBoxTWO.Items.Count);
        }
        else if (length == 3)
        {
            stRead = File.CreateText("Dictionary\\three.txt");
            listBoxTHREE.Items.Remove(textBox3.Text);
            wordTotal = Convert.ToInt32(listBoxTHREE.Items.Count);
        }
        else if (length == 4)
        {
            stRead = File.CreateText("Dictionary\\four.txt");
            listBoxFOUR.Items.Remove(textBox3.Text);
            wordTotal = Convert.ToInt32(listBoxFOUR.Items.Count);
        }

        // Rewrites all words back into the file, except the one to be removed.
        for (listCount = 0; listCount <= wordTotal - 1; listCount++)
        { // For loop to update the respective list of words.
            if ((listCount != 0) && length == 1)
            { // For every loop from listCount == 1 onward, adds each word to the    ↙
file.
                Temp = Temp + System.Environment.NewLine + listBoxONE.Items          ↙
[listCount].ToString();
            }
            else if ((listCount != 0) && length == 2)
            {
                Temp = Temp + System.Environment.NewLine + listBoxTWO.Items          ↙
[listCount].ToString();
            }
            else if ((listCount != 0) && length == 3)
            {
                Temp = Temp + System.Environment.NewLine + listBoxTHREE.Items        ↙
[listCount].ToString();
            }
            else if ((listCount != 0) && length == 4)
            {
                Temp = Temp + System.Environment.NewLine + listBoxFOUR.Items         ↙
[listCount].ToString();
            }
            else
            { // Else statement used to write the initial line to the file.
                if (length == 1)
                {
                    Temp = listBoxONE.Items[0].ToString();
                }
                else if (length == 2)
                {
                    Temp = listBoxTWO.Items[0].ToString();
                }
                else if (length == 3)
                {
                    Temp = listBoxTHREE.Items[0].ToString();
                }
                else if (length == 4)
                {
                    Temp = listBoxFOUR.Items[0].ToString();
                }
            }
            listBox2.Items.Remove(textBox3.Text);

        }

        // Close streamReader to prevent any errors.
        stRead.Write(Temp);
        stRead.Flush();
        stRead.Close();
    }

    public void oneRem(int length)
    {//Method used to remove a word from the dictionary,
     //when only one letter words (alphabet) is loaded.

        //Checks that the desired word exists in the list.
        if (listBox2.Items.Contains(textBox3.Text))
        {//Sets streamWriter to the proper file.

            listBox2.Items.Remove(textBox3.Text);
            StreamWriter stRead;
            int listCount;
            string Temp = "";
            stRead = File.CreateText("Dictionary\\one.txt");

            for (listCount = 0; listCount <= listBox2.Items.Count - 1; listCount++)
            {//Goes through the updated list of words and writes them to the file.
                if ((listCount != 0))
                {//Used for all indices except the first one.
                    Temp = Temp + System.Environment.NewLine + listBox2.Items        ↙
[listCount].ToString();
                }
                else
                {//Used to initialize the string.
                    Temp = listBox2.Items[0].ToString();
                }
            }

            // Updates respective files as necessary and closes stRead.
            stRead.Write(Temp);
            stRead.Flush();
            stRead.Close();
        }
    }

    public void twoRem(int length)
    {//Method used to remove a word from the dictionary,
     //when only two letter words are loaded.

        //Checks that the desired word exists in the list.
        if (listBox2.Items.Contains(textBox3.Text))
        {//Sets streamWriter to the proper file.

            listBox2.Items.Remove(textBox3.Text);
            StreamWriter stRead;
            int listCount;
            string Temp = "";
            stRead = File.CreateText("Dictionary\\two.txt");
            for (listCount = 0; listCount <= listBox2.Items.Count - 1; listCount++)
            {//Goes through the updated list of words and writes them to the file.
                if ((listCount != 0))
                {//Used for all indices except the first one.
                    Temp = Temp + System.Environment.NewLine + listBox2.Items         ↙
[listCount].ToString();
                }
                else
                {//Used to initialize the string.
                    Temp = listBox2.Items[0].ToString();
                }
            }

            // Updates respective files as necessary and closes stRead.
            stRead.Write(Temp);
            stRead.Flush();
            stRead.Close();
        }
    }

    public void threeRem(int length)
    {//Method used to remove a word from the dictionary,
     //when only three letter words are loaded.

        //Checks that the desired word exists in the list.
        if (listBox2.Items.Contains(textBox3.Text))
        {//Sets streamWriter to the proper file.
            listBox2.Items.Remove(textBox3.Text);
            StreamWriter stRead;
            int listCount;
            string Temp = "";
            stRead = File.CreateText("Dictionary\\three.txt");

            for (listCount = 0; listCount <= listBox2.Items.Count - 1; listCount++)
            {//Goes through the updated list of words and writes them to the file.
                if ((listCount != 0))
                {//Used for all indices except the first one.
                    Temp = Temp + System.Environment.NewLine + listBox2.Items         ↙
[listCount].ToString();
                }
                else
                {//Used to initialize the string.
                    Temp = listBox2.Items[0].ToString();
                }
            }

            // Updates respective files as necessary and closes stRead.
            stRead.Write(Temp);
            stRead.Flush();
            stRead.Close();
        }
    }

    public void fourRem(int length)
    {//Method used to remove a word from the dictionary,
     //when only four letter words are loaded.

        //Checks that the desired word exists in the list.
        if (listBox2.Items.Contains(textBox3.Text))
        {//Sets streamWriter to the proper file.
            listBox2.Items.Remove(textBox3.Text);
            StreamWriter stRead;
            int listCount;
            string Temp = "";
            stRead = File.CreateText("Dictionary\\four.txt");

            for (listCount = 0; listCount <= listBox2.Items.Count - 1; listCount++)
            {//Goes through the updated list of words and writes them to the file.
```

```
                    if ((listCount != 0))
                    {//Used for all indices except the first one.
                        Temp = Temp + System.Environment.NewLine + listBox2.Items         ↙
[listCount].ToString();
                    }
                    else
                    {//Used to initialize the string.
                        Temp = listBox2.Items[0].ToString();
                    }
                }

                // Updates respective files as necessary and closes stRead.
                stRead.Write(Temp);
                stRead.Flush();
                stRead.Close();
            }
        }

        public void capRem(int length)
        {//Method used to remove a word from the dictionary,
         //when only words containing capital letters are loaded.

            //Checks that the desired word exists in the list.
            if (listBox2.Items.Contains(textBox3.Text))
            {//Sets streamWriter to the proper file.
                listBox2.Items.Remove(textBox3.Text);
                StreamWriter stRead;
                int listCount;
                string Temp = "";
                stRead = File.CreateText("Dictionary\\capitals.txt");

                for (listCount = 0; listCount <= listBox2.Items.Count - 1; listCount++)
                {//Goes through the updated list of words and writes them to the file.
                    if ((listCount != 0))
                    {//Used for all indices except the first one.
                        Temp = Temp + System.Environment.NewLine + listBox2.Items         ↙
[listCount].ToString();
                    }
                    else
                    {//Used to initialize the string.
                        Temp = listBox2.Items[0].ToString();
                    }
                }

                // Updates respective files as necessary and closes stRead.
                stRead.Write(Temp);
                stRead.Flush();
                stRead.Close();
            }
        }

        public void numRem(int length)
        {//Method used to remove a word from the dictionary,
         //when only numbers loaded.
            int num = Convert.ToInt32(textBox3.Text);
            //Checks that the desired word exists in the list.
            if (listBox2.Items.Contains(num))
            {//Sets streamWriter to the proper file.
                int index = listBox2.Items.IndexOf(num);
                listBox2.Items.RemoveAt(index);
                StreamWriter stRead;
                int listCount;
                string Temp = "";
                stRead = File.CreateText("Dictionary\\numbers.txt");


                for (listCount = 0; listCount <= listBox2.Items.Count - 1; listCount++)
                {//Goes through the updated list of words and writes them to the file.
                    if ((listCount != 0))
                    {//Used for all indices except the first one.
                        Temp = Temp + System.Environment.NewLine + listBox2.Items        ↙
    [listCount].ToString();
                    }
                    else
                    {//Used to initialize the string.
                        Temp = listBox2.Items[0].ToString();
                    }
                }

                // Updates respective files as necessary and closes stRead.
                stRead.Write(Temp);
                stRead.Flush();
                stRead.Close();
            }
        }

        public void checkRem()
        {//Method used to determine which list should be referenced
         //for removing the desired word.

            int listCount = Convert.ToInt32(listBox2.Items.Count);
            int length = textBox3.Text.Length;

            int charCheck = -3;
            charCheck = checkCharacter();
            if (charCheck == 1 && textBoxNUMCH.Text == "1")
            {
                numRem(length);
            }
            else if (textBoxALLCH.Text == "1" && length < 5 && length > 0 && charCheck ==↙
     0)
            {
                allRem(length);
            }
            else if (textBoxONECH.Text == "1" && charCheck == 0)
            {
                oneRem(length);
            }
            else if (textBoxTWOCH.Text == "1" && charCheck == 0)
            {
                twoRem(length);
            }
            else if (textBoxTHREECH.Text == "1" && charCheck == 0)
            {
                threeRem(length);
            }
            else if (textBoxFOURCH.Text == "1" && charCheck == 0)
            {
                fourRem(length);
            }
            else if (textBoxCAPCH.Text == "1" && charCheck == 2)
            {
                capRem(length);
            }
            textBox3.Clear();
        }
        //Methods for removing from Lists end here


        //**Methods for changing Lists start here
        public void changeList()
        {//Method used to load different dictionary lists.

            listBox2.Sorted = true;

            imageBoxClear();
```

```
//Clear out all image boxes.

listBox2.Items.Clear();
string temp = "";
if (listBox3.SelectedIndex == -1)
{//Prevents errors when no list is chosen.

    MessageBox.Show("Error: No list chosen!");
}
else if (listBox3.SelectedItem.ToString() == "One Letter Words")
{ // Loads words with only letter length of one.

    //Clears all lists to prevent misreading data.
    clearLists();
    FileInfo file = new FileInfo("Dictionary\\one.txt");
    StreamReader stRead = file.OpenText();

    while (!stRead.EndOfStream)
    {//Reads line by line and adds each word to the listBox.
        temp = Convert.ToString(stRead.ReadLine());
        listBox2.Items.Add(temp);
        listBoxONE.Items.Add(temp);
    }

    //Closes the streamReader and determines the total # of items added.
    stRead.Close();
    textBoxONE.Text = listBox2.Items.Count.ToString();
    textBoxONECH.Text = "1";
    punctHide();
}
else if (listBox3.SelectedItem.ToString() == "Two Letter Words")
{// Loads words with only letter length of two.

    //Clears all lists to prevent misreading data.
    clearLists();
    FileInfo file = new FileInfo("Dictionary\\two.txt");
    StreamReader stRead = file.OpenText();

    while (!stRead.EndOfStream)
    {//Reads line by line and adds each word to the listBox.
        temp = Convert.ToString(stRead.ReadLine());
        listBox2.Items.Add(temp);
        listBoxTWO.Items.Add(temp);
    }

    //Closes the streamReader and determines the total # of items added.
    stRead.Close();
    textBoxTWO.Text = listBox2.Items.Count.ToString();
    textBoxTWOCH.Text = "1";
    punctHide();
}
else if (listBox3.SelectedItem.ToString() == "Three Letter Words")
{// Loads words with only letter length of three.

    //Clears all lists to prevent misreading data.
    clearLists();
    FileInfo file = new FileInfo("Dictionary\\three.txt");
    StreamReader stRead = file.OpenText();

    while (!stRead.EndOfStream)
    {//Reads line by line and adds each word to the listBox.
        temp = Convert.ToString(stRead.ReadLine());
        listBox2.Items.Add(temp);
        listBoxTHREE.Items.Add(temp);
    }

    //Closes the streamReader and determines the total # of items added.
    stRead.Close();
```

```
        textBoxTHREE.Text = listBox2.Items.Count.ToString();
        textBoxTHREECH.Text = "1";
        punctHide();
    }
    else if (listBox3.SelectedItem.ToString() == "Four Letter Words")
    {// Loads words with only letter length of four.

        //Clears all lists to prevent misreading data.
        clearLists();
        FileInfo file = new FileInfo("Dictionary\\four.txt");
        StreamReader stRead = file.OpenText();

        while (!stRead.EndOfStream)
        {//Reads line by line and adds each word to the listBox.
            temp = Convert.ToString(stRead.ReadLine());
            listBox2.Items.Add(temp);
            listBoxFOUR.Items.Add(temp);
        }

        //Closes the streamReader and determines the total # of items added.
        stRead.Close();
        textBoxFOUR.Text = listBox2.Items.Count.ToString();
        textBoxFOURCH.Text = "1";
        punctHide();
    }
    else if (listBox3.SelectedItem.ToString() == "Capital Letter in Word")
    {// Loads all words length 1-3 that contain a Capital letter.
     // Braille uses one letter to distinguish the following word is capitalized.

        //Clears all lists to prevent misreading data.
        clearLists();
        FileInfo file = new FileInfo("Dictionary\\capitals.txt");
        StreamReader stRead = file.OpenText();

        while (!stRead.EndOfStream)
        {//Reads line by line and adds each word to the listBox.
            temp = Convert.ToString(stRead.ReadLine());
            listBox2.Items.Add(temp);
            listBoxCAP.Items.Add(temp);
        }
        //Closes the streamReader and determines the total # of items added.
        stRead.Close();
        textBoxCAP.Text = listBox2.Items.Count.ToString();
        textBoxCAPCH.Text = "1";
        punctHide();
    }
    else if (listBox3.SelectedItem.ToString() == "All Words")
    {// Loads all words 1-4 length. Does not include
     // capital letters, numbers, punctuation, etc.

        //Clears all lists to prevent misreading data.
        //Prepares to read in words consisting of only 1 letter.
        clearLists();
        FileInfo file = new FileInfo("Dictionary\\one.txt");
        StreamReader stRead = file.OpenText();

        while (!stRead.EndOfStream)
        {//Reads line by line and adds each word to the listBox.
            temp = Convert.ToString(stRead.ReadLine());
            listBox2.Items.Add(temp);
            listBoxONE.Items.Add(temp);
        }

        //Closes stream and tracks total count of 1 letter words.
        stRead.Close();
        textBoxONE.Text = listBox2.Items.Count.ToString();
        int countONE = Convert.ToInt32(textBoxONE.Text);
```

```
            //Prepares to read in words consisting of 2 letters.
            FileInfo file2 = new FileInfo("Dictionary\\two.txt");
            StreamReader stRead2 = file2.OpenText();

            while (!stRead2.EndOfStream)
            {//Reads line by line and adds each word to the listBox.
                temp = Convert.ToString(stRead2.ReadLine());
                listBox2.Items.Add(temp);
                listBoxTWO.Items.Add(temp);
            }

            //Closes stream and tracks total count of 2 letter words.
            stRead2.Close();
            textBoxTWO.Text = listBox2.Items.Count.ToString();
            int countTWO = Convert.ToInt32(textBoxTWO.Text);
            countTWO = countTWO - countONE;
            textBoxTWO.Text = Convert.ToString(countTWO);

            //Prepares to read in words consisting of 3 letters.
            FileInfo file3 = new FileInfo("Dictionary\\three.txt");
            StreamReader stRead3 = file3.OpenText();

            while (!stRead3.EndOfStream)
            {//Reads line by line and adds each word to the listBox
                temp = Convert.ToString(stRead3.ReadLine());
                listBox2.Items.Add(temp);
                listBoxTHREE.Items.Add(temp);
            }

            //Closes stream and tracks total count of 3 letter words.
            stRead3.Close();
            textBoxTHREE.Text = listBox2.Items.Count.ToString();
            int countTHREE = Convert.ToInt32(textBoxTHREE.Text);
            countTHREE = countTHREE - countTWO - countONE;
            textBoxTHREE.Text = Convert.ToString(countTHREE);

            //Prepares to read in words consisting of 4 letters.
            FileInfo file4 = new FileInfo("Dictionary\\four.txt");
            StreamReader stRead4 = file4.OpenText();

            while (!stRead4.EndOfStream)
            {//Reads line by line and adds each word to the listBox
                temp = Convert.ToString(stRead4.ReadLine());
                listBox2.Items.Add(temp);
                listBoxFOUR.Items.Add(temp);
            }

            //Closes stream and tracks total count of 4 letter words.
            stRead4.Close();
            textBoxFOUR.Text = listBox2.Items.Count.ToString();
            int countFOUR = Convert.ToInt32(textBoxFOUR.Text);
            countFOUR = countFOUR - countTHREE - countTWO - countONE;
            textBoxFOUR.Text = Convert.ToString(countFOUR);

            //Tracks total number of words overall.
            textBoxALL.Text = listBox2.Items.Count.ToString();
            textBoxALLCH.Text = "1";
            punctHide();
        }
        else if (listBox3.SelectedItem.ToString() == "Numbers")
        {
            listBox2.Sorted = false;

            clearLists();
            FileInfo file = new FileInfo("Dictionary\\numbers.txt");
            StreamReader stRead = file.OpenText();
            while (!stRead.EndOfStream)
            {
                temp = Convert.ToString(stRead.ReadLine());
                listBox2.Items.Add(temp);
                listBoxNUM.Items.Add(temp);
            }
            stRead.Close();
            textBoxNUM.Text = listBox2.Items.Count.ToString();
            textBoxNUMCH.Text = "1";
            punctHide();

            numericList();
        }
        else if (listBox3.SelectedItem.ToString() == "Punctuation")
        {
            clearLists();
            FileInfo file = new FileInfo("Dictionary\\punctuation.txt");
            StreamReader stRead = file.OpenText();
            while (!stRead.EndOfStream)
            {
                temp = Convert.ToString(stRead.ReadLine());
                listBox2.Items.Add(temp);
                listBoxPUN.Items.Add(temp);
            }
            stRead.Close();
            textBoxPUN.Text = listBox2.Items.Count.ToString();
            textBoxPUNCH.Text = "1";
            punctHide();
        }
    }

    public void clearLists()
    {//Method used to clear every listbox and textbox
     //used in tracking words/counts/which list is
     //being read from.

        listBoxONE.Items.Clear();
        listBoxTWO.Items.Clear();
        listBoxTHREE.Items.Clear();
        listBoxFOUR.Items.Clear();
        listBoxNUM.Items.Clear();
        listBoxPUN.Items.Clear();
        listBoxCAP.Items.Clear();
        listBoxALL.Items.Clear();
        listBoxRIGHT.Items.Clear();
        listBoxWRONG.Items.Clear();
        listBoxTest.Items.Clear();
        // Clears all lists to prevent incorrect access

        textBoxONE.Clear();
        textBoxTWO.Clear();
        textBoxTHREE.Clear();
        textBoxFOUR.Clear();
        textBoxNUM.Clear();
        textBoxPUN.Clear();
        textBoxCAP.Clear();
        // Clears all textboxes used to track word counts.

        textBoxALL.Clear();

        textBoxONECH.Clear();
        textBoxTWOCH.Clear();
        textBoxTHREECH.Clear();
        textBoxFOURCH.Clear();
        textBoxALLCH.Clear();
        textBoxNUMCH.Clear();
        textBoxPUNCH.Clear();
        textBoxCAPCH.Clear();
        textBoxMatINPUT.Clear();
        textBoxMatOUTPUT.Clear();
        // Clears all checks used to track which list is being used.

        pictureBox24.Image = null;
        pictureBox25.Image = null;
        pictureBox26.Image = null;
        pictureBox27.Image = null;
        pictureBox28.Image = null;
        pictureBox29.Image = null;
        pictureBox30.Image = null;
        pictureBox31.Image = null;

        testString = "";
        guessString = "";
        indexString[0] = 0;
        indexString[1] = 0;
        indexString[2] = 0;
        indexString[3] = 0;
        testCounter = 0;
        maxTemp = 0;
        label43.Text = "0";
        // Clears all variables used for tests.

    }

    private void listPopup()
    {//Method used to change GUI for changing between
     //dictionary lists.

        if (panelLearn.Visible)
        {
            panelLearn.Hide();
            panelImage.BringToFront();
            button17.Show();
        }

        if (button29.Text.ToString() == "Choose Word Selection")
        {//Pressing "Change Word Selection" will expand the menu
         //and show all available dictionary lists.

            //Change GUI to hide menu on next press.
            button29.Text = "Hide Word Selection";
            listBox3.Show();
            button30.Show();
            if (listBox3.Items.Contains("Punctuation"))
            {
                // Do nothing. Handled elsewhere.
            }
            else
            {//Reads in list of all available files.

                FileInfo file = new FileInfo("Files\\Files.txt");
                StreamReader stRead = file.OpenText();
                while (!stRead.EndOfStream)
                {
                    listBox3.Items.Add(stRead.ReadLine());
                }
                stRead.Close();
            }
        }
        else
        {//Pressing "Hide List Selection" will cause the
         //menu to collapse and hide all options.

            button29.Text = "Choose Word Selection";
            listBox3.Hide();
            button30.Hide();
        }
    }
```

```
private void loadTestWords()
{//Method used to determine which test (random/timed)
 //mode was chosen and calls the function to generate
 //a set of words.

    string temp = "";
    if (textBoxONECH.Text == "1")
    {//Implies 1 letter words were chosen in the test menu.

        //Sets streamReader to the proper file.
        clearLists();
        FileInfo file = new FileInfo("Dictionary\\one.txt");
        StreamReader stRead = file.OpenText();

        while (!stRead.EndOfStream)
        {//Reads in all available words under that list.
            temp = Convert.ToString(stRead.ReadLine());
            listBoxONE.Items.Add(temp);
        }

        // Closes stRead and tracks total number of words in list.
        stRead.Close();
        textBoxONE.Text = listBoxONE.Items.Count.ToString();
        textBoxONECH.Text = "1";

        //Checks to see if mode is Timed or Random and calls
        //the proper function.
        if (textBoxTestTimer.Text == "1")
        {
            testAdd(1);
        }
        else if (textBoxRandom.Text == "1")
        {
            randomAdd(1);
        }
    }
    else if (textBoxTWOCH.Text == "1")
    {//Implies 2 letter words were chosen in the test menu.

        //Sets streamReader to the proper file.
        clearLists();
        FileInfo file = new FileInfo("Dictionary\\two.txt");
        StreamReader stRead = file.OpenText();

        while (!stRead.EndOfStream)
        {//Reads in all available words under that list.
            temp = Convert.ToString(stRead.ReadLine());
            listBoxTWO.Items.Add(temp);
        }

        // Closes stRead and tracks total number of words in list.
        stRead.Close();
        textBoxTWO.Text = listBoxTWO.Items.Count.ToString();
        textBoxTWOCH.Text = "1";

        //Checks to see if mode is Timed or Random and calls
        //the proper function.
        if (textBoxTestTimer.Text == "1")
        {
            testAdd(2);
        }
        else if (textBoxRandom.Text == "1")
        {
            randomAdd(2);
        }
    }
    else if (textBoxTHREECH.Text == "1")
```

```
{//Implies 3 letter words were chosen in the test menu.

    //Sets streamReader to the proper file.
    clearLists();
    FileInfo file = new FileInfo("Dictionary\\three.txt");
    StreamReader stRead = file.OpenText();

    while (!stRead.EndOfStream)
    {//Reads in all available words under that list.
        temp = Convert.ToString(stRead.ReadLine());
        listBoxTHREE.Items.Add(temp);
    }

    // Closes stRead and tracks total number of words in list.
    stRead.Close();
    textBoxTHREE.Text = listBoxTHREE.Items.Count.ToString();
    textBoxTHREE.Text = "1";

    //Checks to see if mode is Timed or Random and calls
    //the proper function.
    if (textBoxTestTimer.Text == "1")
    {
        testAdd(3);
    }
    else if (textBoxRandom.Text == "1")
    {
        randomAdd(3);
    }
}
else if (textBoxFOURCH.Text == "1")
{//Implies 4 letter words were chosen in the test menu.

    //Sets streamReader to the proper file.
    clearLists();
    FileInfo file = new FileInfo("Dictionary\\four.txt");
    StreamReader stRead = file.OpenText();

    while (!stRead.EndOfStream)
    {//Reads in all available words under that list.
        temp = Convert.ToString(stRead.ReadLine());
        listBoxFOUR.Items.Add(temp);
    }

    // Closes stRead and tracks total number of words in list.
    stRead.Close();
    textBoxFOUR.Text = listBoxFOUR.Items.Count.ToString();
    textBoxFOUR.Text = "1";

    //Checks to see if mode is Timed or Random and calls
    //the proper function.
    if (textBoxTestTimer.Text == "1")
    {
        testAdd(4);
    }
    else if (textBoxRandom.Text == "1")
    {
        randomAdd(4);
    }
}
else if (textBoxALLCH.Text == "1")
{//Implies 1-4 letter words were chosen in the test menu.
 //Does not include capital letters or numbers.

    //Sets streamReader to read in 1 letter words.
    clearLists();
    FileInfo file = new FileInfo("Dictionary\\one.txt");
    StreamReader stRead = file.OpenText();
```

```
        while (!stRead.EndOfStream)
        {//Reads in all available words under that list.
            temp = Convert.ToString(stRead.ReadLine());
            listBoxALL.Items.Add(temp);
        }
        stRead.Close();

        //Sets streamReader to read in 2 letter words.
        FileInfo file2 = new FileInfo("Dictionary\\two.txt");
        StreamReader stRead2 = file2.OpenText();

        while (!stRead2.EndOfStream)
        {//Reads in all available words under that list.
            temp = Convert.ToString(stRead2.ReadLine());
            listBoxALL.Items.Add(temp);
        }
        stRead2.Close();

        //Sets streamReader to read in 3 letter words.
        FileInfo file3 = new FileInfo("Dictionary\\three.txt");
        StreamReader stRead3 = file3.OpenText();

        while (!stRead3.EndOfStream)
        {//Reads in all available words under that list.
            temp = Convert.ToString(stRead3.ReadLine());
            listBoxALL.Items.Add(temp);
        }
        stRead3.Close();

        //Sets streamReader to read in 4 letter words.
        FileInfo file4 = new FileInfo("Dictionary\\four.txt");
        StreamReader stRead4 = file4.OpenText();

        while (!stRead4.EndOfStream)
        {//Reads in all available words under that list.
            temp = Convert.ToString(stRead4.ReadLine());
            listBoxALL.Items.Add(temp);
        }
        stRead4.Close();

        //Tracks total number of words in the list.
        //Tracks that all words are being used for random generation
        textBoxALL.Text = listBox2.Items.Count.ToString();
        textBoxALLCH.Text = "1";

        //Checks to see if mode is Timed or Random and calls
        //the proper function.
        if (textBoxTestTimer.Text == "1")
        {
            testAdd(5);
        }
        else if (textBoxRandom.Text == "1")
        {
            randomAdd(5);
        }
    }

    else if (textBoxNUMCH.Text == "1")
    {//Implies numbers were chosen in the test menu.

        //Sets streamReader to the proper file.
        clearLists();
        FileInfo file = new FileInfo("Dictionary\\numbers.txt");
        StreamReader stRead = file.OpenText();

        while (!stRead.EndOfStream)
        {//Reads in all available words under that list.
            temp = Convert.ToString(stRead.ReadLine());
```

```
                listBoxNUM.Items.Add(temp);
            }

            // Closes stRead and tracks total number of numberical words in list.
            stRead.Close();
            textBoxNUM.Text = listBoxNUM.Items.Count.ToString();
            textBoxNUMCH.Text = "1";

            //Checks to see if mode is Timed or Random and calls
            //the proper function.
            if (textBoxTestTimer.Text == "1")
            {
                testAdd(6);
            }
            else if (textBoxRandom.Text == "1")
            {
                randomAdd(6);
            }
        }

    }

    private void punctHide()
    {// Hides add/remove word menus. Prevents changing list
     // of words when punctuation is chosen.

        if (textBoxPUNCH.Text == "1")
        {
            button17.Text = "Modify List";
            button17.Hide();
            textBox3.Hide();
            button19.Hide();
            button31.Hide();
        }
        else
        {
            button17.Show();
        }
    }
    //Methods for changing Lists end here

    //** Methods used for testing purposes start here
    private bool checkCapital(ListBox listName)
    {//Method used to check if incoming word contains a capital letter.

        // Creates a string consisting of every letter AFTER the first letter
        string temp = listName.SelectedItem.ToString();
        string subtemp = temp.Substring(1);
        string firstLetter = Convert.ToString(temp[0]);

        if (System.Text.RegularExpressions.Regex.IsMatch(firstLetter, "[A-Z]"))
        { // Check that the first letter is capital only
            if (System.Text.RegularExpressions.Regex.IsMatch(subtemp, "[a-z]"))
            {
                return true;
            }
            else if (subtemp == "")
            {
                return true;
            }
            return false;
        }
        else
        {
            return false;
        }

    }
    private int checkCharacter()
    {//Method used to check if word contains proper characters.
     //ONLY USED WHEN ADDING OR REMOVING a word.

        if (textBox3.Text == "")
        { // Empty text field.
            return -2;
        }
        else
        {//Determines characters and length of word.

            int check = 0;
            string temp = textBox3.Text.ToString();
            string subtemp = "";

            if (temp.Length > 1)
            {//Checks if word has 2 or more characters.
             //Used to handle cases with capital letters.

                subtemp = temp.Substring(1);
            }

            string firstLetter = Convert.ToString(temp[0]);

            if (System.Text.RegularExpressions.Regex.IsMatch(temp, "^[0-9]"))
            {// Checks if word to be added is strictly a number.
                if (Convert.ToInt32(textBox3.Text) > 999)
                {// Checks if word to be added contains anything NON-lowercase
                    // Words consisting of Capital letters or numerical numbers are  ↙
flagged.
                    check = -3;
                }
                else
                {
                   check = 1;
                }
            }
            else if (System.Text.RegularExpressions.Regex.IsMatch(firstLetter, "[A-Z]↙
"))
            { // Check that the first letter is capital only
                if (System.Text.RegularExpressions.Regex.IsMatch(subtemp, "[a-z]"))
                {
                    check = 2;
                }
                else if (subtemp == "")
                { // Case of adding or removing a single letter Capital word.
                    check = 2;
                }
            }
            else if (System.Text.RegularExpressions.Regex.IsMatch(textBox3.Text, "[^a↙
-z]"))
            {// Checks if word to be added contains anything NON-lowercase
                // Words consisting of Capital letters or numerical numbers are      ↙
flagged.
                check = -1;
            }
            else
            {
                check = 0;
            }

            return check;
        }
    }

    private int checkNum(ListBox listName)
    {//Method used to check if incoming word is strictly numerical.
```

```
        int check = 0;
        string temp = listName.SelectedItem.ToString();
        string subtemp = temp.Substring(1);
        string firstLetter = Convert.ToString(temp[0]);

        if (System.Text.RegularExpressions.Regex.IsMatch(temp, "^[0-9]"))
        {// Checks if word to be added is strictly a number.
         //Implies word is strictly numerical.

            check = 1;
        }
        else
        {//Implies word is not strictly numerical.

            check = 0;
        }
        return check;
    }

    private void chooseWord(int val)
    {//Method used to determine word type, to send to microcontroller
     //and display the proper pictureBoxes on screen.

        if (listBox2.SelectedIndex == -1)
        {//Prevents sending when no word is chosen.
            MessageBox.Show("Error: No word chosen!");
        }
        else
        {
            if (textBoxNUMCH.Text == "1")
            {//Checks if list is strictly numerical.

                loadImageNum(val);
            }
            else if (textBoxPUNCH.Text == "1")
            {//Checks if image is strictly punctuation.

                loadImagePunc();
            }
            else
            {//Handles all other cases.

                loadImage(val);
            }
        }
    }

    private void listBoxRIGHT_SelectedIndexChanged(object sender, EventArgs e)
    {//Event handler used in TEST/RANDOM modes for Right/Wrong word listBoxes
     //Prevents selecting an index in both listboxes.

        if (checkToggle == 1)
        {//Temporarily prevents listBoxWRONG_SelectedIndexChanged from firing.

            checkToggle = 0;
            listBoxWRONG.SelectedIndex = -1;
            checkToggle = 1;
        }
    }

    private void listBoxWRONG_SelectedIndexChanged(object sender, EventArgs e)
    {//Event handler used in TEST/RANDOM modes for Right/Wrong word listBoxes
     //Prevents selecting an index in both listboxes.

        if (checkToggle == 1)
        {//Temporarily prevents listBoxRIGHT_SelectedIndexChanged from firing.
```

```
                checkToggle = 0;
                listBoxRIGHT.SelectedIndex = -1;
                checkToggle = 1;
            }
        }

        private void matlabINPUT()
        {//Tracks the current letter used in Speech Recognition
         //Used to know what values to send to MATLAB function.

            if (listBoxTest.Items.Count != 0)
            {
                if (listBoxTest.SelectedItem.ToString() == "a")
                {
                    textBoxMatINPUT.Text = "1";
                }
                else if (listBoxTest.SelectedItem.ToString() == "b")
                {
                    textBoxMatINPUT.Text = "2";
                }
                else if (listBoxTest.SelectedItem.ToString() == "c")
                {
                    textBoxMatINPUT.Text = "3";
                }
                else if (listBoxTest.SelectedItem.ToString() == "d")
                {
                    textBoxMatINPUT.Text = "4";
                }
                else if (listBoxTest.SelectedItem.ToString() == "e")
                {
                    textBoxMatINPUT.Text = "5";
                }
                else if (listBoxTest.SelectedItem.ToString() == "f")
                {
                    textBoxMatINPUT.Text = "6";
                }
                else if (listBoxTest.SelectedItem.ToString() == "g")
                {
                    textBoxMatINPUT.Text = "7";
                }
                else if (listBoxTest.SelectedItem.ToString() == "h")
                {
                    textBoxMatINPUT.Text = "8";
                }
                else if (listBoxTest.SelectedItem.ToString() == "i")
                {
                    textBoxMatINPUT.Text = "9";
                }
                else if (listBoxTest.SelectedItem.ToString() == "j")
                {
                    textBoxMatINPUT.Text = "10";
                }
                else if (listBoxTest.SelectedItem.ToString() == "k")
                {
                    textBoxMatINPUT.Text = "11";
                }
                else if (listBoxTest.SelectedItem.ToString() == "l")
                {
                    textBoxMatINPUT.Text = "12";
                }
                else if (listBoxTest.SelectedItem.ToString() == "m")
                {
                    textBoxMatINPUT.Text = "13";
                }
                else if (listBoxTest.SelectedItem.ToString() == "n")
                {
                    textBoxMatINPUT.Text = "14";
                }
                else if (listBoxTest.SelectedItem.ToString() == "o")
                {
                    textBoxMatINPUT.Text = "15";
                }
                else if (listBoxTest.SelectedItem.ToString() == "p")
                {
                    textBoxMatINPUT.Text = "16";
                }
                else if (listBoxTest.SelectedItem.ToString() == "q")
                {
                    textBoxMatINPUT.Text = "17";
                }
                else if (listBoxTest.SelectedItem.ToString() == "r")
                {
                    textBoxMatINPUT.Text = "18";
                }
                else if (listBoxTest.SelectedItem.ToString() == "s")
                {
                    textBoxMatINPUT.Text = "19";
                }
                else if (listBoxTest.SelectedItem.ToString() == "t")
                {
                    textBoxMatINPUT.Text = "20";
                }
                else if (listBoxTest.SelectedItem.ToString() == "u")
                {
                    textBoxMatINPUT.Text = "21";
                }
                else if (listBoxTest.SelectedItem.ToString() == "v")
                {
                    textBoxMatINPUT.Text = "22";
                }
                else if (listBoxTest.SelectedItem.ToString() == "w")
                {
                    textBoxMatINPUT.Text = "23";
                }
                else if (listBoxTest.SelectedItem.ToString() == "x")
                {
                    textBoxMatINPUT.Text = "24";
                }
                else if (listBoxTest.SelectedItem.ToString() == "y")
                {
                    textBoxMatINPUT.Text = "25";
                }
                else if (listBoxTest.SelectedItem.ToString() == "z")
                {
                    textBoxMatINPUT.Text = "26";
                }
            }
            else
            {
                //Do nothing.
            }
        }

        private void panelTestDone()
        {//Method used when either Timed/Random tests have been
         //completed. Will not show if menu if test is stopped or exited.

            panelTest1.Hide();
            panelTest5.Hide();
            panelTest3.Hide();
            panelTest7.Hide();
            panelTestCancel.Hide();
            //Hide the panels to send words/change guessed letters.

            panelNav1.Show();
            panelNav1.BringToFront();
            panelTest4.Show();
            panelTestHidden.Show();

            //Show an escape button, lists of both correct and skipped words.
        }

        //Empty method. Need to program.
        private void playSound(string soundFile)
        {//Method used to play sounds.
            if (soundFile == "a")
            {
                soundByte = new SoundPlayer(Properties.Resources._10);
            }
            else if (soundFile == "b")
            {
                soundByte = new SoundPlayer(Properties.Resources._20);
            }
            else if (soundFile == "c")
            {
                soundByte = new SoundPlayer(Properties.Resources._30);
            }
            else if (soundFile == "d")
            {
                soundByte = new SoundPlayer(Properties.Resources._40);
            }
            else if (soundFile == "e")
            {
                soundByte = new SoundPlayer(Properties.Resources._50);
            }
            else if (soundFile == "f")
            {
                soundByte = new SoundPlayer(Properties.Resources._60);
            }
            else if (soundFile == "g")
            {
                soundByte = new SoundPlayer(Properties.Resources._70);
            }
            else if (soundFile == "h")
            {
                soundByte = new SoundPlayer(Properties.Resources._80);
            }
            else if (soundFile == "i")
            {
                soundByte = new SoundPlayer(Properties.Resources._90);
            }
            else if (soundFile == "j")
            {
                soundByte = new SoundPlayer(Properties.Resources._100);
            }
            else if (soundFile == "k")
            {
                soundByte = new SoundPlayer(Properties.Resources._110);
            }
            else if (soundFile == "l")
            {
                soundByte = new SoundPlayer(Properties.Resources._120);
            }
            else if (soundFile == "m")
            {
                soundByte = new SoundPlayer(Properties.Resources._130);
            }
            else if (soundFile == "n")
            {
                soundByte = new SoundPlayer(Properties.Resources._140);
            }
            else if (soundFile == "o")
            {
                soundByte = new SoundPlayer(Properties.Resources._150);
            }
            else if (soundFile == "p")
            {
                soundByte = new SoundPlayer(Properties.Resources._160);
            }
            else if (soundFile == "q")
            {
                soundByte = new SoundPlayer(Properties.Resources._170);
            }
            else if (soundFile == "r")
            {
                soundByte = new SoundPlayer(Properties.Resources._180);
            }
            else if (soundFile == "s")
            {
                soundByte = new SoundPlayer(Properties.Resources._190);
            }
            else if (soundFile == "t")
            {
                soundByte = new SoundPlayer(Properties.Resources._200);
            }
            else if (soundFile == "u")
            {
                soundByte = new SoundPlayer(Properties.Resources._210);
            }
            else if (soundFile == "v")
            {
                soundByte = new SoundPlayer(Properties.Resources._220);
            }
            else if (soundFile == "w")
            {
                soundByte = new SoundPlayer(Properties.Resources._230);
            }
            else if (soundFile == "x")
            {
                soundByte = new SoundPlayer(Properties.Resources._240);
            }
            else if (soundFile == "y")
            {
                soundByte = new SoundPlayer(Properties.Resources._250);
            }
            else if (soundFile == "z")
            {
                soundByte = new SoundPlayer(Properties.Resources._260);
            }
        }

        private void testRight()
        {//Method used while in Test/Random Mode.
         //Implies word has been correctly guessed and will load the next word.

            int index = listBoxTest.SelectedIndex;

            if (index < listBoxTest.Items.Count - 1)
            {//Implies that this IS NOT the last word in the list.

                //Copy correct word to listboxRIGHT then move to the next index.
                listBoxRIGHT.Items.Add(listBoxTest.Items[index]);
                index++;
                listBoxTest.SelectedIndex = index;

                // Clear all used variables and set testString to the current word.
                testString = listBoxTest.Text;
                guessString = "";
                indexString[0] = 0;
                indexString[1] = 0;
                indexString[2] = 0;
                indexString[3] = 0;
```

```
                testCounter++;
                label43.Text = Convert.ToString(testCounter);

                // Use a notifier that the word was guessed properly and that a new word ↙
        is coming.
                MessageBox.Show("Correctly identified word. Next word incoming...");

                if (panelTest2.Visible)
                {//Update stats for Timed Test panel
                    statsCurrTest[0]++;
                    statsCurrTest[2]++;
                    statsTestUpdate();
                }
                else if (panelRandStats.Visible)
                {//Update stats for Random test panel
                    statsCurrRand[0]++;
                    statsCurrRand[2]++;
                    statsRandUpdate();
                }

                testWord();
                microTest();
            }
            else if (index == listBoxTest.Items.Count - 1)
            {//Implies that this IS the last word in the list.

                //Copy correct word to listboxRIGHT then move to the next index.
                listBoxRIGHT.Items.Add(listBoxTest.Items[index]);

                // Clear all used variables and set testString to the current word.
                testString = "";
                guessString = "";
                indexString[0] = 0;
                indexString[1] = 0;
                indexString[2] = 0;
                indexString[3] = 0;

                panelTestDone();
                timerReset();
                testCounter++;
                label43.Text = Convert.ToString(testCounter);

                if (panelTest2.Visible)
                {//Update stats for Timed Test panel
                    statsCurrTest[0]++;
                    statsCurrTest[2]++;
                    statsTestUpdate();
                    statsTestRec();
                }
                else if (panelRandStats.Visible)
                {//Update stats for Random test panel
                    statsCurrRand[0]++;
                    statsCurrRand[2]++;
                    statsRandUpdate();
                    statsRandRec();
                }
                if (textBoxNUMCH.Text == "1")
                {
                    numericRight();
                    numericWrong();
                }
                MessageBox.Show("Test completed.");
            }
        }

        public void testTog(Button buttonName)
        {//Used to toggle visibility for a button that is passed.
```

```
 //Only used for testing/debugging.

    if (buttonName.Visible)
    {
        buttonName.Hide();
    }
    else
    {
        buttonName.Show();
    }
}

public void testTog2(ListBox listName)
{//Used to toggle visibility for a listbox that is passed.
 //Only used for testing/debugging.

    if (listName.Visible)
    {
        listName.Hide();
    }
    else
    {
        listName.Show();
    }
}

private void testWord()
{//Method used to load a word for testing in
 //either Test or Random Mode. Can be called
 //after correctly identifying a word, choosing to skip
 //or at the start of the test.

 //**ONLY USED TO LOAD IMAGES.** Microcontroller data is handled
 //in another function.

    if (listBoxTest.SelectedIndex == -1)
    {//Implies no word is chosen in the list and error handles.
        MessageBox.Show("Error: No word chosen!");
    }
    else
    {//Implies a word has been chosen and determines the proper
     //method to call to load the proper picture boxes.

        if (textBoxNUMCH.Text == "1")
        {//Implies word is strictly numerical.
            loadTestImageNum();
        }
        else if (textBoxPUNCH.Text == "1")
        {//Implies word is strictly punctuation
            loadTestImagePunc();
        }
        else
        {//Implies word is strictly letters.
            loadTestImage();
        }
    }
}

private void testWrong()
{//Method used while in Test/Random Mode. Skips the current word.
 //Implies word has been INCORRECTLY guessed and will load the next word.

    textBox1.Text = "";
    textBox4.Text = "";

    int index = listBoxTest.SelectedIndex;

    if (index < listBoxTest.Items.Count - 1)
```

```
        {
            //Copy correct word to listboxWRONG then move to the next index.
            listBoxWRONG.Items.Add(listBoxTest.Items[index]);
            index++;
            listBoxTest.SelectedIndex = index;

            // Clear all used variables and set testString to the current word.
            testString = listBoxTest.Text;
            guessString = "";
            indexString[0] = 0;
            indexString[1] = 0;
            indexString[2] = 0;
            indexString[3] = 0;

            testCounter++;
            label43.Text = Convert.ToString(testCounter);

            if (panelTest2.Visible)
            {//Update stats for Timed Test panel
                statsCurrTest[1]++;
                statsCurrTest[2]++;
                statsTestUpdate();
            }
            else if (panelRandStats.Visible)
            {//Update stats for Random test panel
                statsCurrRand[1]++;
                statsCurrRand[2]++;
                statsRandUpdate();
            }

            // Use a notifier that the word was guessed properly and that a new word ↙
is coming.
            MessageBox.Show("Word skipped. Next word incoming...");


            testWord();
            microTest();

        }
        else if (index == listBoxTest.Items.Count - 1)
        {
            listBoxWRONG.Items.Add(listBoxTest.Items[index]);
            //Copy correct word to listboxRIGHT then move to the next index.

            testString = "";
            guessString = "";
            indexString[0] = 0;
            indexString[1] = 0;
            indexString[2] = 0;
            indexString[3] = 0;
            // Clear all used variables and set testString to the current word.
            panelTestDone();
            timerReset();
            testCounter++;
            label43.Text = Convert.ToString(testCounter);

            if (panelTest2.Visible)
            {//Update stats for Timed Test panel
                statsCurrTest[1]++;
                statsCurrTest[2]++;
                statsTestUpdate();
                statsTestRec();
            }
            else if (panelRandStats.Visible)
            {//Update stats for Random test panel
                statsCurrRand[1]++;
                statsCurrRand[2]++;
                statsRandUpdate();
```

```
                    statsRandRec();
                }
                if (textBoxNUMCH.Text == "1")
                {
                    numericRight();
                    numericWrong();
                }
                MessageBox.Show("Test completed.");
            }
        }

        private void randWord()
        {//Method used to randomly choose from the list.
         //Used in Learning Mode.

            if (listBox2.Items.Count == 0)
            {//Error handling if no words are in list.
                MessageBox.Show("No words in list to choose from!");
            }
            else
            {//Randomly generates a number in the threshold and chooses
             //that index for the random word.

                Random objRan = new Random();
                int rand = objRan.Next(0, listBox2.Items.Count);
                // Range from index 0 to listBox.Items.Count
                // Max value is listBox.Items.Count because it never actually reaches    ↙
    that #.
                listBox2.SelectedIndex = rand;
            }
        }
        //Methods used for testing purposes end here


        //**Methods for updating images start here
        private void loadTestImage()
        {//Method used to determine which pictureBoxes are to have
         //images loaded into them.
         //ONLY USED IN TESTING MODES, WHERE "engWord" MATTERS.

            // engWord = 0 will not show any English letters.
            // engWord = 1 will show the entire word in English.
            // engWord = 2 will only show correctly guessed letters.

            //Starts at pictureBox28 (right-most Braille pictBox).
            int pictCount = 28;
            PictureBox pictBr = null;
            PictureBox pictEng = null;
            PictureBox pictBr2 = null;
            PictureBox pictEng2 = null;
            int checkNumerical = -3;
            bool checkChar;

            if (listBoxTest.SelectedIndex == -1)
            {
                // Do nothing.
            }
            else
            {
                //Calls functions to check for special cases.
                checkNumerical = checkNum(listBoxTest);
                checkChar = checkCapital(listBoxTest);

                // Takes selected item in listbox and converts to a string variable.
                string temp = listBoxTest.SelectedItem.ToString();

                int length = temp.Length; ;
                int count = length - 1;
```

```
char letter = ' ';

while (count != -1)
{
    if (checkChar == false || count != 0)
    {// checkChar == false implies CURRENT letter isn't capitalized.
        if (pictCount == 28)
        {//Deals with right-most pictureBox.

            //Clears out ALL Braille pictureBoxes.
            pictureBox28.Image = null;
            pictureBox28.Invalidate();

            pictureBox29.Image = null;
            pictureBox29.Invalidate();

            pictureBox30.Image = null;
            pictureBox30.Invalidate();

            pictureBox31.Image = null;
            pictureBox31.Invalidate();

            // Clear ALL English picture boxes before continuing.
            pictureBox24.Image = null;
            pictureBox24.Invalidate();

            pictureBox25.Image = null;
            pictureBox25.Invalidate();

            pictureBox26.Image = null;
            pictureBox26.Invalidate();

            pictureBox27.Image = null;
            pictureBox27.Invalidate();


            pictBr = pictureBox28;
            pictEng = pictureBox24;
        }
        else if (pictCount == 29)
        {//Deals with 2nd right-most pictureBox.
            pictBr = pictureBox29;
            pictEng = pictureBox25;
        }
        else if (pictCount == 30)
        {//Deals with 3rd right-most pictureBox.
            pictBr = pictureBox30;
            pictEng = pictureBox26;
        }
        else if (pictCount == 31)
        {//Deals with 4th right-most pictureBox.
         //(AKA the left-most pictureBox.)
            pictBr = pictureBox31;
            pictEng = pictureBox27;
        }

        // Converts the selected string index to a character.
        letter = Convert.ToChar(temp[count]);

        // Sends the letter to be loaded to the proper pictureBox.
        letterTestImage(letter, pictBr, pictEng, pictCount);

        // Work in reverse order, start by loading the rightmost letter
        // and work towards loading the first letter last.
        count--;

        // Move to the next pictureBox (ie. from right to left).
        pictCount++;
```

```
                }
                else
                { // Means letter contains a capital letter. Handles capital letters
                  //only. All other letters are handled above.

                    if (pictCount == 28)
                    {
                        // Clear all Braille picture boxes before continuing.
                        pictureBox28.Image = null;
                        pictureBox28.Invalidate();

                        pictureBox29.Image = null;
                        pictureBox29.Invalidate();

                        pictureBox30.Image = null;
                        pictureBox30.Invalidate();

                        pictureBox31.Image = null;
                        pictureBox31.Invalidate();

                        // Clear all English picture boxes before continuing.
                        pictureBox24.Image = null;
                        pictureBox24.Invalidate();

                        pictureBox25.Image = null;
                        pictureBox25.Invalidate();

                        pictureBox26.Image = null;
                        pictureBox26.Invalidate();

                        pictureBox27.Image = null;
                        pictureBox27.Invalidate();

                        // Deals with 2 right-most pictureBoxes in both English/     ↙
Braille.

                        pictBr = pictureBox28;
                        pictBr2 = pictureBox29;
                        pictEng = pictureBox24;
                        pictEng2 = pictureBox25;
                    }
                    else if (pictCount == 29)
                    {//Deals with 3rd and 2nd right-most pictureBoxes.
                        pictBr = pictureBox29;
                        pictBr2 = pictureBox30;
                        pictEng = pictureBox25;
                        pictEng2 = pictureBox26;
                    }
                    else if (pictCount == 30)
                    {//Deals with 4th and 3rd right-most pictureBox.
                     //(AKA two left-most pictureBoxes)

                        pictBr = pictureBox30;
                        pictBr2 = pictureBox31;
                        pictEng = pictureBox26;
                        pictEng2 = pictureBox27;
                    }
                    else
                    {
                        //Do nothing.
                    }

                    // Converts the selected string index to a character.
                    letter = Convert.ToChar(temp[count]);

                    // Sends the letter to be loaded to the proper pictureBox.
                    letterTestImageCap(letter, pictBr, pictBr2, pictEng, pictEng2,   ↙
pictCount);
```

```
                // Work in reverse order, start by loading the rightmost letter
                // and work towards loading the first letter last.
                count--;

                // Move to the next pictureBox (ie. from right to left).
                pictCount++;

            }
        }
    }
}

private void loadTestImageNum()
{//Method used to determine which pictureBoxes are to have
 //images loaded into them. Used to load numberical words into pictureBoxes.

    int pictCount = 28;
    PictureBox pictBr = null;
    PictureBox pictEng = null;
    PictureBox pictBr2 = null;
    PictureBox pictEng2 = null;

    // Used to check if the string contains a capital letter.
    bool check;

    if (listBoxTest.SelectedIndex == -1)
    {
        // Do nothing, no index selected.
    }
    else
    {
        // Takes selected item in listbox and converts to a string variable.
        check = checkCapital(listBoxTest);
        string temp = listBoxTest.SelectedItem.ToString();

        int length = temp.Length;
        int count = length - 1;
        char letter = ' ';

        while (count != -1)
        {
            if (count != 0)
            {
                if (pictCount == 28)
                {
                    // Clear all Braille picture boxes before continuing.
                    pictureBox28.Image = null;
                    pictureBox28.Invalidate();

                    pictureBox29.Image = null;
                    pictureBox29.Invalidate();

                    pictureBox30.Image = null;
                    pictureBox30.Invalidate();

                    pictureBox31.Image = null;
                    pictureBox31.Invalidate();

                    // Clear all English picture boxes before continuing.
                    pictureBox24.Image = null;
                    pictureBox24.Invalidate();

                    pictureBox25.Image = null;
                    pictureBox25.Invalidate();

                    pictureBox26.Image = null;
                    pictureBox26.Invalidate();
```

```
                        pictureBox27.Image = null;
                        pictureBox27.Invalidate();

                        // Prepare to change right-most pictureBoxes.
                        pictBr = pictureBox28;
                        pictEng = pictureBox24;
                    }
                    else if (pictCount == 29)
                    {//Deals with 2nd right-most pictureBoxes.

                        pictBr = pictureBox29;
                        pictEng = pictureBox25;
                    }
                    else if (pictCount == 30)
                    {//Deals with 3rd right-most pictureBoxes.

                        pictBr = pictureBox30;
                        pictEng = pictureBox26;
                    }
                    else if (pictCount == 31)
                    {//Deals with 4th right-most pictureBoxes.
                     //(AKA left-most pictureBox)

                        pictBr = pictureBox31;
                        pictEng = pictureBox27;
                    }

                    letter = Convert.ToChar(temp[count]);
                    // Converts the selected string index to a character.
                    numTestImage(letter, pictBr, pictEng, pictCount);
                    // Sends the letter to be loaded to the proper pictureBox.
                    count--;
                    // Work in reverse order, start by loading the rightmost letter
                    // and work towards loading the first letter last.
                    pictCount++;
                    // Move to the next pictureBox (ie. from right to left).
                }
                else
                { // Means first digit in word (left-most), must add in Braille      ↙
symbol for indicating a number.
                    if (pictCount == 28)
                    {
                        // Clear all Braille picture boxes before continuing.
                        pictureBox28.Image = null;
                        pictureBox28.Invalidate();

                        pictureBox29.Image = null;
                        pictureBox29.Invalidate();

                        pictureBox30.Image = null;
                        pictureBox30.Invalidate();

                        pictureBox31.Image = null;
                        pictureBox31.Invalidate();

                        // Clear all English picture boxes before continuing.
                        pictureBox24.Image = null;
                        pictureBox24.Invalidate();

                        pictureBox25.Image = null;
                        pictureBox25.Invalidate();

                        pictureBox26.Image = null;
                        pictureBox26.Invalidate();

                        pictureBox27.Image = null;
                        pictureBox27.Invalidate();
```

```
                                //Deals with 2nd and 1st right-most pictureBoxes.
                                pictBr = pictureBox28;
                                pictBr2 = pictureBox29;
                                pictEng = pictureBox24;
                                pictEng2 = pictureBox25;
                            }
                            else if (pictCount == 29)
                            {//Deals with 3rd and 2nd right-most pictureBoxes.
                                pictBr = pictureBox29;
                                pictBr2 = pictureBox30;
                                pictEng = pictureBox25;
                                pictEng2 = pictureBox26;
                            }
                            else if (pictCount == 30)
                            {//Deals with 4th and 3rd right-most pictureBoxes.
                             //(AKA 2 left-most pictureBoxes)

                                pictBr = pictureBox30;
                                pictBr2 = pictureBox31;
                                pictEng = pictureBox26;
                                pictEng2 = pictureBox27;
                            }
                            else
                            {
                                //Do nothing.
                            }

                            // Converts the selected string index to a character.
                            letter = Convert.ToChar(temp[count]);

                            // Sends the letter to be loaded to the proper pictureBox.
                            firstTestDigitImage(letter, pictBr, pictBr2, pictEng, pictEng2,  ↙
        pictCount);

                            // Work in reverse order, start by loading the rightmost letter
                            // and work towards loading the first letter last.
                            count--;

                            // Move to the next pictureBox (ie. from right to left).
                            pictCount++;
                        }
                    }
                }
            }

            private void loadTestImagePunc()
            {//Method used to determine which pictureBoxes are to have
             //images loaded into them when punctuation is chosen.
             //Punctuation is only one character long.

                // Clear all Braille picture boxes before continuing.
                pictureBox28.Image = null;
                pictureBox28.Invalidate();

                pictureBox29.Image = null;
                pictureBox29.Invalidate();

                pictureBox30.Image = null;
                pictureBox30.Invalidate();

                pictureBox31.Image = null;
                pictureBox31.Invalidate();

                // Clear all English picture boxes before continuing.
                pictureBox24.Image = null;
                pictureBox24.Invalidate();

                pictureBox25.Image = null;
                pictureBox25.Invalidate();

                pictureBox26.Image = null;
                pictureBox26.Invalidate();

                pictureBox27.Image = null;
                pictureBox27.Invalidate();

                //Calls method that loads images into pictureBoxes.
                letterImagePunc(pictureBox28, pictureBox24, listBoxTest);
            }

            private void loadImage(int pictCount)
            {//Method used to determine which pictureBoxes are to have
             //images loaded into them. Primarily called from Learning mode.

                //Remove any references of the following variables:
                PictureBox pictBr = null;
                PictureBox pictEng = null;
                PictureBox pictBr2 = null;
                PictureBox pictEng2 = null;

                int checkNumerical = -3;
                bool checkChar;

                if (listBox2.SelectedIndex == -1)
                {
                   // Do nothing. Me
                }
                else
                {
                    checkNumerical = checkNum(listBox2);
                    checkChar = checkCapital(listBox2);

                    // Takes selected item in listbox and converts to a string variable.
                    string temp = listBox2.SelectedItem.ToString();

                    int length = temp.Length; ;
                    int count = length - 1;
                    char letter = ' ';

                    while (count != -1)
                    {
                        if (checkChar == false || count != 0)
                        {// Check == false implies no capital letter.
                            if (pictCount == 28)
                            {
                                // Clear all Braille picture boxes before continuing.
                                pictureBox28.Image = null;
                                pictureBox28.Invalidate();

                                pictureBox29.Image = null;
                                pictureBox29.Invalidate();

                                pictureBox30.Image = null;
                                pictureBox30.Invalidate();

                                pictureBox31.Image = null;
                                pictureBox31.Invalidate();

                                // Clear all English picture boxes before continuing.
                                pictureBox24.Image = null;
                                pictureBox24.Invalidate();

                                pictureBox25.Image = null;
                                pictureBox25.Invalidate();

                                pictureBox26.Image = null;
                    pictureBox26.Invalidate();

                    pictureBox27.Image = null;
                    pictureBox27.Invalidate();

                    //Deals with right-most pictureBoxes in both Braille/English.
                    pictBr = pictureBox28;
                    pictEng = pictureBox24;
                }
                else if (pictCount == 29)
                {//Deals with 2nd right-most pictureBoxes.
                    pictBr = pictureBox29;
                    pictEng = pictureBox25;
                }
                else if (pictCount == 30)
                {//Deals with 3rd right-most pictureBoxes.
                    pictBr = pictureBox30;
                    pictEng = pictureBox26;
                }
                else if (pictCount == 31)
                {//Deals with 4th right-most pictureBoxes.
                 //(AKA left-most pictureBox)
                    pictBr = pictureBox31;
                    pictEng = pictureBox27;
                }

                // Converts the selected string index to a character.
                letter = Convert.ToChar(temp[count]);

                // Sends the letter to be loaded to the proper pictureBox.
                letterImage(letter, pictBr, pictEng);

                // Work in reverse order, start by loading the rightmost letter
                // and work towards loading the first letter last.
                count--;

                // Move to the next pictureBox (ie. from right to left).
                pictCount++;
            }
            else
            { // Means letter contains a capital letter.
                if (pictCount == 28)
                {
                    pictureBox28.Image = null;
                    pictureBox28.Invalidate();

                    pictureBox29.Image = null;
                    pictureBox29.Invalidate();

                    pictureBox30.Image = null;
                    pictureBox30.Invalidate();

                    pictureBox31.Image = null;
                    pictureBox31.Invalidate();
                    // Clear all Braille picture boxes before continuing.

                    pictureBox24.Image = null;
                    pictureBox24.Invalidate();

                    pictureBox25.Image = null;
                    pictureBox25.Invalidate();

                    pictureBox26.Image = null;
                    pictureBox26.Invalidate();

                    pictureBox27.Image = null;
                    pictureBox27.Invalidate();
                    // Clear all English picture boxes before continuing.
```

```
                        //Deals with 2nd and 1st right-most pictureBoxes.
                        pictBr = pictureBox28;
                        pictBr2 = pictureBox29;
                        pictEng = pictureBox24;
                        pictEng2 = pictureBox25;
                    }
                    else if (pictCount == 29)
                    {//Deals with 3rd and 2nd right-most pictureBoxes.
                        pictBr = pictureBox29;
                        pictBr2 = pictureBox30;
                        pictEng = pictureBox25;
                        pictEng2 = pictureBox26;
                    }
                    else if (pictCount == 30)
                    {//Deals with 4th and 3rd right-most pictureBoxes.
                     //(AKA 2 left-most pictureBoxes)
                        pictBr = pictureBox30;
                        pictBr2 = pictureBox31;
                        pictEng = pictureBox26;
                        pictEng2 = pictureBox27;
                    }
                    else
                    {
                        //Do Nothing.
                    }

                    // Converts the selected string index to a character.
                    letter = Convert.ToChar(temp[count]);

                    // Sends the letter to be loaded to the proper pictureBox.
                    letterImageCap(letter, pictBr, pictBr2, pictEng, pictEng2);

                    // Work in reverse order, start by loading the rightmost letter
                    // and work towards loading the first letter last.
                    count--;

                    // Move to the next pictureBox (ie. from right to left).
                    pictCount++;
                }
            }
        }
    }

    private void loadImageNum(int pictCount)
    {//Method used to determine which pictureBoxes are to have
     //images loaded into them.

        //Dereference the following variables.
        PictureBox pictBr = null;
        PictureBox pictEng = null;
        PictureBox pictBr2 = null;
        PictureBox pictEng2 = null;

        // Used to check if the string contains a capital letter.
        bool check;


        if (listBox2.SelectedIndex == -1)
        {
            // Do nothing. Me
        }
        else
        {
            check = checkCapital(listBox2);
            string temp = listBox2.SelectedItem.ToString();
            // Takes selected item in listbox and converts to a string variable.
            int length = temp.Length; ;
```

```
int count = length - 1;
char letter = ' ';

while (count != -1)
{
    if (count != 0)
    {
        if (pictCount == 28)
        {
            // Clear all Braille picture boxes before continuing.
            pictureBox28.Image = null;
            pictureBox28.Invalidate();

            pictureBox29.Image = null;
            pictureBox29.Invalidate();

            pictureBox30.Image = null;
            pictureBox30.Invalidate();

            pictureBox31.Image = null;
            pictureBox31.Invalidate();

            // Clear all English picture boxes before continuing.
            pictureBox24.Image = null;
            pictureBox24.Invalidate();

            pictureBox25.Image = null;
            pictureBox25.Invalidate();

            pictureBox26.Image = null;
            pictureBox26.Invalidate();

            pictureBox27.Image = null;
            pictureBox27.Invalidate();

            //Deals with right-most pictureBox.
            pictBr = pictureBox28;
            pictEng = pictureBox24;
        }
        else if (pictCount == 29)
        {//Deals with 2nd right-most pictureBox.
            pictBr = pictureBox29;
            pictEng = pictureBox25;
        }
        else if (pictCount == 30)
        {//Deals with 3rd right-most pictureBox.
            pictBr = pictureBox30;
            pictEng = pictureBox26;
        }
        else if (pictCount == 31)
        {//Deals with 4th right-most pictureBox.
         // (AKA left-most pictureBox)

            pictBr = pictureBox31;
            pictEng = pictureBox27;
        }

        letter = Convert.ToChar(temp[count]);
        // Converts the selected string index to a character.
        numImage(letter, pictBr, pictEng);
        // Sends the letter to be loaded to the proper pictureBox.
        count--;
        // Work in reverse order, start by loading the rightmost letter
        // and work towards loading the first letter last.
        pictCount++;
        // Move to the next pictureBox (ie. from right to left).
    }
    else
```

```
                { // Means first digit (left-most), must also add in Braille symbol  ↙
for indicating a number.
                    if (pictCount == 28)
                    {
                        // Clear all Braille picture boxes before continuing.
                        pictureBox28.Image = null;
                        pictureBox28.Invalidate();

                        pictureBox29.Image = null;
                        pictureBox29.Invalidate();

                        pictureBox30.Image = null;
                        pictureBox30.Invalidate();

                        pictureBox31.Image = null;
                        pictureBox31.Invalidate();

                        // Clear all English picture boxes before continuing.
                        pictureBox24.Image = null;
                        pictureBox24.Invalidate();

                        pictureBox25.Image = null;
                        pictureBox25.Invalidate();

                        pictureBox26.Image = null;
                        pictureBox26.Invalidate();

                        pictureBox27.Image = null;
                        pictureBox27.Invalidate();

                        //Deals with 2nd and 1st right-most pictureBoxes.
                        pictBr = pictureBox28;
                        pictBr2 = pictureBox29;
                        pictEng = pictureBox24;
                        pictEng2 = pictureBox25;
                    }
                    else if (pictCount == 29)
                    {//Deals with 3rd and 2nd right-most pictureBoxes.
                        pictBr = pictureBox29;
                        pictBr2 = pictureBox30;
                        pictEng = pictureBox25;
                        pictEng2 = pictureBox26;
                    }
                    else if (pictCount == 30)
                    {//Deals with 4th and 3rd right-most pictureBoxes.
                     //(AKA 2 left-most pictureBoxes)
                        pictBr = pictureBox30;
                        pictBr2 = pictureBox31;
                        pictEng = pictureBox26;
                        pictEng2 = pictureBox27;
                    }
                    else
                    {
                        //Do nothing.
                    }
                    // Converts the selected string index to a character.
                    letter = Convert.ToChar(temp[count]);

                    // Sends the letter to be loaded to the proper pictureBox.
                    firstDigitImage(letter, pictBr, pictBr2, pictEng, pictEng2);

                    // Work in reverse order, start by loading the rightmost letter
                    // and work towards loading the first letter last.
                    count--;

                    // Move to the next pictureBox (ie. from right to left).
                    pictCount++;
                }
```

```
            }
        }
    }

    private void loadImagePunc()
    {//Method used to choose which pictureBoxes
     //to load their respective images with.

        // Clear all Braille picture boxes before continuing.
        pictureBox28.Image = null;
        pictureBox28.Invalidate();

        pictureBox29.Image = null;
        pictureBox29.Invalidate();

        pictureBox30.Image = null;
        pictureBox30.Invalidate();

        pictureBox31.Image = null;
        pictureBox31.Invalidate();

        // Clear all English picture boxes before continuing.
        pictureBox24.Image = null;
        pictureBox24.Invalidate();

        pictureBox25.Image = null;
        pictureBox25.Invalidate();

        pictureBox26.Image = null;
        pictureBox26.Invalidate();

        pictureBox27.Image = null;
        pictureBox27.Invalidate();

        //Calls method that loads the images.
        letterImagePunc(pictureBox28, pictureBox24, listBox2);
    }

    private void letterTestImage(char letter, PictureBox pictBr, PictureBox pictEng, ↙
int pictCount)
    {//Method used to load images into the chosen pictureBoxes.
     //Used in either Test or Random mode.

        //Dependant on engWord to show English pictureBoxes.
        // engWord = 0 will not show any English letters.
        // engWord = 1 will show the entire word in English.
        // engWord = 2 will only show correctly guessed letters.

        if (letter == 'a')
        {
            pictBr.Image = Properties.Resources.A_br;
            pictEng.Image = Properties.Resources.A_eng;
        }
        else if (letter == 'b')
        {
            pictBr.Image = Properties.Resources.B_br;
            pictEng.Image = Properties.Resources.B_eng;
        }
        else if (letter == 'c')
        {
            pictBr.Image = Properties.Resources.C_br;
            pictEng.Image = Properties.Resources.C_eng;
        }
        else if (letter == 'd')
        {
            pictBr.Image = Properties.Resources.D_br;
            pictEng.Image = Properties.Resources.D_eng;
        }
```

```
else if (letter == 'e')
{
    pictBr.Image = Properties.Resources.E_br;
    pictEng.Image = Properties.Resources.E_eng;
}
else if (letter == 'f')
{
    pictBr.Image = Properties.Resources.F_br;
    pictEng.Image = Properties.Resources.F_eng;
}
else if (letter == 'g')
{
    pictBr.Image = Properties.Resources.G_br;
    pictEng.Image = Properties.Resources.G_eng;
}
else if (letter == 'h')
{
    pictBr.Image = Properties.Resources.H_br;
    pictEng.Image = Properties.Resources.H_eng;
}
else if (letter == 'i')
{
    pictBr.Image = Properties.Resources.I_br;
    pictEng.Image = Properties.Resources.I_eng;
}
else if (letter == 'j')
{
    pictBr.Image = Properties.Resources.J_br;
    pictEng.Image = Properties.Resources.J_eng;
}
else if (letter == 'k')
{
    pictBr.Image = Properties.Resources.K_br;
    pictEng.Image = Properties.Resources.K_eng;
}
else if (letter == 'l')
{
    pictBr.Image = Properties.Resources.L_br;
    pictEng.Image = Properties.Resources.L_eng;
}
else if (letter == 'm')
{
    pictBr.Image = Properties.Resources.M_br;
    pictEng.Image = Properties.Resources.M_eng;
}
else if (letter == 'n')
{
    pictBr.Image = Properties.Resources.N_br;
    pictEng.Image = Properties.Resources.N_eng;
}
else if (letter == 'o')
{
    pictBr.Image = Properties.Resources.O_br;
    pictEng.Image = Properties.Resources.O_eng;
}
else if (letter == 'p')
{
    pictBr.Image = Properties.Resources.P_br;
    pictEng.Image = Properties.Resources.P_eng;
}
else if (letter == 'q')
{
    pictBr.Image = Properties.Resources.Q_br;
    pictEng.Image = Properties.Resources.Q_eng;
}
else if (letter == 'r')
{
    pictBr.Image = Properties.Resources.R_br;
```

```
                pictEng.Image = Properties.Resources.R_eng;
            }
            else if (letter == 's')
            {
                pictBr.Image = Properties.Resources.S_br;
                pictEng.Image = Properties.Resources.S_eng;
            }
            else if (letter == 't')
            {
                pictBr.Image = Properties.Resources.T_br;
                pictEng.Image = Properties.Resources.T_eng;
            }
            else if (letter == 'u')
            {
                pictBr.Image = Properties.Resources.U_br;
                pictEng.Image = Properties.Resources.U_eng;
            }
            else if (letter == 'v')
            {
                pictBr.Image = Properties.Resources.V_br;
                pictEng.Image = Properties.Resources.V_eng;
            }
            else if (letter == 'w')
            {
                pictBr.Image = Properties.Resources.W_br;
                pictEng.Image = Properties.Resources.W_eng;
            }
            else if (letter == 'x')
            {
                pictBr.Image = Properties.Resources.X_br;
                pictEng.Image = Properties.Resources.X_eng;
            }
            else if (letter == 'y')
            {
                pictBr.Image = Properties.Resources.Y_br;
                pictEng.Image = Properties.Resources.Y_eng;
            }
            else if (letter == 'z')
            {
                pictBr.Image = Properties.Resources.Z_br;
                pictEng.Image = Properties.Resources.Z_eng;
            }

            if (engWord == 0)
            {// Means word is to be hidden. Load a "Question Mark" into pictureBox.
                pictEng.Image = Properties.Resources.QuestionMark_eng;
            }
            else if (engWord == 1)
            {// Means full word is to be shown.
            }
            else if (engWord == 2)
            {// Means only correctly guessed letters are to be shown.

                if (indexString[28- pictCount + testString.Length -1] == 1)
                {//pictureBox increment goes from right to left, string increment goes     ↙
        from left to right.
                    // Image is already properly loaded
                }
                else
                { // Letter is incorrectly guessed, load question mark.
                    pictEng.Image = Properties.Resources.QuestionMark_eng;
                }
            }

        }

        private void letterTestImageCap(char letter, PictureBox pictBr, PictureBox       ↙
        pictBr2, PictureBox pictEng, PictureBox pictEng2, int pictCount)
```

```
{//Method used to load images into the chosen pictureBoxes.
 //Used in either Test/Random Mode.
 //Deals with capital letters.
 //Reminder: Upper case notation requires two letters in Braille.

    //Dependant on engWord to show English pictureBoxes.
    // engWord = 0 will not show any English letters.
    // engWord = 1 will show the entire word in English.
    // engWord = 2 will only show correctly guessed letters.

    if (letter == 'A')
    {
        pictBr.Image = Properties.Resources.A_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capA_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'B')
    {
        pictBr.Image = Properties.Resources.B_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capB_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'C')
    {
        pictBr.Image = Properties.Resources.C_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capC_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'D')
    {
        pictBr.Image = Properties.Resources.D_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capD_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'E')
    {
        pictBr.Image = Properties.Resources.E_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capE_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'F')
    {
        pictBr.Image = Properties.Resources.F_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capF_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'G')
    {
        pictBr.Image = Properties.Resources.G_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capG_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'H')
    {
        pictBr.Image = Properties.Resources.H_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capH_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'I')
    {
```

```
    pictBr.Image = Properties.Resources.I_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capI_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'J')
{
    pictBr.Image = Properties.Resources.J_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capJ_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'K')
{
    pictBr.Image = Properties.Resources.K_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capK_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'L')
{
    pictBr.Image = Properties.Resources.L_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capL_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'M')
{
    pictBr.Image = Properties.Resources.M_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capM_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'N')
{
    pictBr.Image = Properties.Resources.N_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capN_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'O')
{
    pictBr.Image = Properties.Resources.O_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capO_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'P')
{
    pictBr.Image = Properties.Resources.P_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capP_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'Q')
{
    pictBr.Image = Properties.Resources.Q_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capQ_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'R')
{
    pictBr.Image = Properties.Resources.R_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capR_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
```

```
        else if (letter == 'S')
        {
            pictBr.Image = Properties.Resources.S_br;
            pictBr2.Image = Properties.Resources.CapitalSign_br;
            pictEng.Image = Properties.Resources.capS_eng;
            pictEng2.Image = Properties.Resources.CapitalSign_eng;
        }
        else if (letter == 'T')
        {
            pictBr.Image = Properties.Resources.T_br;
            pictBr2.Image = Properties.Resources.CapitalSign_br;
            pictEng.Image = Properties.Resources.capT_eng;
            pictEng2.Image = Properties.Resources.CapitalSign_eng;
        }
        else if (letter == 'U')
        {
            pictBr.Image = Properties.Resources.U_br;
            pictBr2.Image = Properties.Resources.CapitalSign_br;
            pictEng.Image = Properties.Resources.capU_eng;
            pictEng2.Image = Properties.Resources.CapitalSign_eng;
        }
        else if (letter == 'V')
        {
            pictBr.Image = Properties.Resources.V_br;
            pictBr2.Image = Properties.Resources.CapitalSign_br;
            pictEng.Image = Properties.Resources.capV_eng;
            pictEng2.Image = Properties.Resources.CapitalSign_eng;
        }
        else if (letter == 'W')
        {
            pictBr.Image = Properties.Resources.W_br;
            pictBr2.Image = Properties.Resources.CapitalSign_br;
            pictEng.Image = Properties.Resources.capW_eng;
            pictEng2.Image = Properties.Resources.CapitalSign_eng;
        }
        else if (letter == 'X')
        {
            pictBr.Image = Properties.Resources.X_br;
            pictBr2.Image = Properties.Resources.CapitalSign_br;
            pictEng.Image = Properties.Resources.capX_eng;
            pictEng2.Image = Properties.Resources.CapitalSign_eng;
        }
        else if (letter == 'Y')
        {
            pictBr.Image = Properties.Resources.Y_br;
            pictBr2.Image = Properties.Resources.CapitalSign_br;
            pictEng.Image = Properties.Resources.capY_eng;
            pictEng2.Image = Properties.Resources.CapitalSign_eng;
        }
        else if (letter == 'Z')
        {
            pictBr.Image = Properties.Resources.Z_br;
            pictBr2.Image = Properties.Resources.CapitalSign_br;
            pictEng.Image = Properties.Resources.capZ_eng;
            pictEng2.Image = Properties.Resources.CapitalSign_eng;
        }
        if (engWord == 0)
        {// Means word is to be hidden. Load a "Question Mark" into pictureBox.
            pictEng.Image = Properties.Resources.QuestionMark_eng;
        }
        else if (engWord == 1)
        {// Means full word is to be shown.
        }
        else if (engWord == 2)
        {// Means only correctly guessed letters are to be shown.
            if (indexString[28 - pictCount + testString.Length - 1] == 1)
            {//pictureBox increment goes from right to left, string increment goes  ↙
from left to right.
```

```
                    // Image is already properly loaded
                }
                else
                { // Letter is incorrectly guessed, load question mark.
                    pictEng.Image = Properties.Resources.QuestionMark_eng;
                }
            }
        }

        private void numTestImage(char letter, PictureBox pictBr, PictureBox pictEng, int↙
     pictCount)
        {//Method used to load images into the chosen pictureBoxes.
         //Used in either Test/Random mode.

         //Dependant on engWord to show English pictureBoxes.
         // engWord = 0 will not show any English letters.
         // engWord = 1 will show the entire word in English.
         // engWord = 2 will only show correctly guessed letters.

         //Reminder: Numbers require an extra letter in Braille to denote it is numerical↙
    .
            if (letter == '1')
            {
                pictBr.Image = Properties.Resources.A_br;
                pictEng.Image = Properties.Resources._1_eng;
            }
            else if (letter == '2')
            {
                pictBr.Image = Properties.Resources.B_br;
                pictEng.Image = Properties.Resources._2_eng;
            }
            else if (letter == '3')
            {
                pictBr.Image = Properties.Resources.C_br;
                pictEng.Image = Properties.Resources._3_eng;
            }
            else if (letter == '4')
            {
                pictBr.Image = Properties.Resources.D_br;
                pictEng.Image = Properties.Resources._4_eng;
            }
            else if (letter == '5')
            {
                pictBr.Image = Properties.Resources.E_br;
                pictEng.Image = Properties.Resources._5_eng;
            }
            else if (letter == '6')
            {
                pictBr.Image = Properties.Resources.F_br;
                pictEng.Image = Properties.Resources._6_eng;
            }
            else if (letter == '7')
            {
                pictBr.Image = Properties.Resources.G_br;
                pictEng.Image = Properties.Resources._7_eng;
            }
            else if (letter == '8')
            {
                pictBr.Image = Properties.Resources.H_br;
                pictEng.Image = Properties.Resources._8_eng;
            }
            else if (letter == '9')
            {
                pictBr.Image = Properties.Resources.I_br;
                pictEng.Image = Properties.Resources._9_eng;
            }
            else if (letter == '0')
            {
```

```
            pictBr.Image = Properties.Resources.J_br;
            pictEng.Image = Properties.Resources._0_eng;
        }

        if (engWord == 0)
        {// Means word is to be hidden. Load a "Question Mark" into pictureBox.
            pictEng.Image = Properties.Resources.QuestionMark_eng;
        }
        else if (engWord == 1)
        {// Means full word is to be shown.
        }
        else if (engWord == 2)
        {// Means only correctly guessed letters are to be shown.
            if (indexString[28 - pictCount + testString.Length - 1] == 1)
            {//pictureBox increment goes from right to left, string increment goes    ↙
from left to right.
                // Image is already properly loaded
            }
            else
            { // Letter is incorrectly guessed, load question mark.
                pictEng.Image = Properties.Resources.QuestionMark_eng;
            }
        }
    }

    private void firstTestDigitImage(char letter, PictureBox pictBr, PictureBox      ↙
pictBr2, PictureBox pictEng, PictureBox pictEng2, int pictCount)
    {//Method used to load images into the chosen pictureBoxes.
     // Method used to deal with the first digit (left-most) in a number
     // in Random/Test mode.

     // Reminder: Upper case notation requires two letters in Braille.

        if (letter == '1')
        {
            pictBr.Image = Properties.Resources.A_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._1_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '2')
        {
            pictBr.Image = Properties.Resources.B_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._2_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '3')
        {
            pictBr.Image = Properties.Resources.C_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._3_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '4')
        {
            pictBr.Image = Properties.Resources.D_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._4_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '5')
        {
            pictBr.Image = Properties.Resources.E_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._5_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '6')
        {
            pictBr.Image = Properties.Resources.F_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._6_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '7')
        {
            pictBr.Image = Properties.Resources.G_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._7_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '8')
        {
            pictBr.Image = Properties.Resources.H_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._8_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '9')
        {
            pictBr.Image = Properties.Resources.I_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._9_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '0')
        {
            pictBr.Image = Properties.Resources.J_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._0_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }

        if (engWord == 0)
        {// Means word is to be hidden. Load a "Question Mark" into pictureBox.
            pictEng.Image = Properties.Resources.QuestionMark_eng;
        }
        else if (engWord == 1)
        {// Means full word is to be shown.
        }
        else if (engWord == 2)
        {// Means only correctly guessed letters are to be shown.
            if (indexString[28 - pictCount + testString.Length - 1] == 1)
            {//pictureBox increment goes from right to left, string increment goes   ↙
from left to right.
                // Image is already properly loaded
            }
            else
            { // Letter is incorrectly guessed, load question mark.
                pictEng.Image = Properties.Resources.QuestionMark_eng;
            }
        }
    }

    private void letterImage(char letter, PictureBox pictBr, PictureBox pictEng)
    {//Method used to load images into the chosen pictureBoxes.
     //Used in Learning Mode, where all letters are shown.

        if (letter == 'a')
        {
            pictBr.Image = Properties.Resources.A_br;
            pictEng.Image = Properties.Resources.A_eng;
        }
        else if (letter == 'b')
        {
```

```
    pictBr.Image = Properties.Resources.B_br;
    pictEng.Image = Properties.Resources.B_eng;
}
else if (letter == 'c')
{
    pictBr.Image = Properties.Resources.C_br;
    pictEng.Image = Properties.Resources.C_eng;
}
else if (letter == 'd')
{
    pictBr.Image = Properties.Resources.D_br;
    pictEng.Image = Properties.Resources.D_eng;
}
else if (letter == 'e')
{
    pictBr.Image = Properties.Resources.E_br;
    pictEng.Image = Properties.Resources.E_eng;
}
else if (letter == 'f')
{
    pictBr.Image = Properties.Resources.F_br;
    pictEng.Image = Properties.Resources.F_eng;
}
else if (letter == 'g')
{
    pictBr.Image = Properties.Resources.G_br;
    pictEng.Image = Properties.Resources.G_eng;
}
else if (letter == 'h')
{
    pictBr.Image = Properties.Resources.H_br;
    pictEng.Image = Properties.Resources.H_eng;
}
else if (letter == 'i')
{
    pictBr.Image = Properties.Resources.I_br;
    pictEng.Image = Properties.Resources.I_eng;
}
else if (letter == 'j')
{
    pictBr.Image = Properties.Resources.J_br;
    pictEng.Image = Properties.Resources.J_eng;
}
else if (letter == 'k')
{
    pictBr.Image = Properties.Resources.K_br;
    pictEng.Image = Properties.Resources.K_eng;
}
else if (letter == 'l')
{
    pictBr.Image = Properties.Resources.L_br;
    pictEng.Image = Properties.Resources.L_eng;
}
else if (letter == 'm')
{
    pictBr.Image = Properties.Resources.M_br;
    pictEng.Image = Properties.Resources.M_eng;
}
else if (letter == 'n')
{
    pictBr.Image = Properties.Resources.N_br;
    pictEng.Image = Properties.Resources.N_eng;
}
else if (letter == 'o')
{
    pictBr.Image = Properties.Resources.O_br;
    pictEng.Image = Properties.Resources.O_eng;
}
```

```
            else if (letter == 'p')
            {
                pictBr.Image = Properties.Resources.P_br;
                pictEng.Image = Properties.Resources.P_eng;
            }
            else if (letter == 'q')
            {
                pictBr.Image = Properties.Resources.Q_br;
                pictEng.Image = Properties.Resources.Q_eng;
            }
            else if (letter == 'r')
            {
                pictBr.Image = Properties.Resources.R_br;
                pictEng.Image = Properties.Resources.R_eng;
            }
            else if (letter == 's')
            {
                pictBr.Image = Properties.Resources.S_br;
                pictEng.Image = Properties.Resources.S_eng;
            }
            else if (letter == 't')
            {
                pictBr.Image = Properties.Resources.T_br;
                pictEng.Image = Properties.Resources.T_eng;
            }
            else if (letter == 'u')
            {
                pictBr.Image = Properties.Resources.U_br;
                pictEng.Image = Properties.Resources.U_eng;
            }
            else if (letter == 'v')
            {
                pictBr.Image = Properties.Resources.V_br;
                pictEng.Image = Properties.Resources.V_eng;
            }
            else if (letter == 'w')
            {
                pictBr.Image = Properties.Resources.W_br;
                pictEng.Image = Properties.Resources.W_eng;
            }
            else if (letter == 'x')
            {
                pictBr.Image = Properties.Resources.X_br;
                pictEng.Image = Properties.Resources.X_eng;
            }
            else if (letter == 'y')
            {
                pictBr.Image = Properties.Resources.Y_br;
                pictEng.Image = Properties.Resources.Y_eng;
            }
            else if (letter == 'z')
            {
                pictBr.Image = Properties.Resources.Z_br;
                pictEng.Image = Properties.Resources.Z_eng;
            }
        }

        private void letterImageCap(char letter, PictureBox pictBr, PictureBox pictBr2,  ↙
    PictureBox pictEng, PictureBox pictEng2)
        {//Method used to load images into the chosen pictureBoxes.
         //Used in Learning Mode where all letters are displayed.

         // Method used to deal with capital letters.
         // Upper case notation requires two letters in Braille.

            if (letter == 'A')
            {
                pictBr.Image = Properties.Resources.A_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capA_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'B')
    {
        pictBr.Image = Properties.Resources.B_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capB_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'C')
    {
        pictBr.Image = Properties.Resources.C_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capC_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'D')
    {
        pictBr.Image = Properties.Resources.D_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capD_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'E')
    {
        pictBr.Image = Properties.Resources.E_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capE_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'F')
    {
        pictBr.Image = Properties.Resources.F_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capF_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'G')
    {
        pictBr.Image = Properties.Resources.G_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capG_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'H')
    {
        pictBr.Image = Properties.Resources.H_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capH_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'I')
    {
        pictBr.Image = Properties.Resources.I_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capI_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'J')
    {
        pictBr.Image = Properties.Resources.J_br;
        pictBr2.Image = Properties.Resources.CapitalSign_br;
        pictEng.Image = Properties.Resources.capJ_eng;
        pictEng2.Image = Properties.Resources.CapitalSign_eng;
    }
    else if (letter == 'K')
```

```
{
    pictBr.Image = Properties.Resources.K_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capK_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'L')
{
    pictBr.Image = Properties.Resources.L_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capL_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'M')
{
    pictBr.Image = Properties.Resources.M_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capM_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'N')
{
    pictBr.Image = Properties.Resources.N_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capN_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'O')
{
    pictBr.Image = Properties.Resources.O_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capO_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'P')
{
    pictBr.Image = Properties.Resources.P_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capP_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'Q')
{
    pictBr.Image = Properties.Resources.Q_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capQ_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'R')
{
    pictBr.Image = Properties.Resources.R_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capR_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'S')
{
    pictBr.Image = Properties.Resources.S_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capS_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
}
else if (letter == 'T')
{
    pictBr.Image = Properties.Resources.T_br;
    pictBr2.Image = Properties.Resources.CapitalSign_br;
    pictEng.Image = Properties.Resources.capT_eng;
    pictEng2.Image = Properties.Resources.CapitalSign_eng;
```

```
            }
            else if (letter == 'U')
            {
                pictBr.Image = Properties.Resources.U_br;
                pictBr2.Image = Properties.Resources.CapitalSign_br;
                pictEng.Image = Properties.Resources.capU_eng;
                pictEng2.Image = Properties.Resources.CapitalSign_eng;
            }
            else if (letter == 'V')
            {
                pictBr.Image = Properties.Resources.V_br;
                pictBr2.Image = Properties.Resources.CapitalSign_br;
                pictEng.Image = Properties.Resources.capV_eng;
                pictEng2.Image = Properties.Resources.CapitalSign_eng;
            }
            else if (letter == 'W')
            {
                pictBr.Image = Properties.Resources.W_br;
                pictBr2.Image = Properties.Resources.CapitalSign_br;
                pictEng.Image = Properties.Resources.capW_eng;
                pictEng2.Image = Properties.Resources.CapitalSign_eng;
            }
            else if (letter == 'X')
            {
                pictBr.Image = Properties.Resources.X_br;
                pictBr2.Image = Properties.Resources.CapitalSign_br;
                pictEng.Image = Properties.Resources.capX_eng;
                pictEng2.Image = Properties.Resources.CapitalSign_eng;
            }
            else if (letter == 'Y')
            {
                pictBr.Image = Properties.Resources.Y_br;
                pictBr2.Image = Properties.Resources.CapitalSign_br;
                pictEng.Image = Properties.Resources.capY_eng;
                pictEng2.Image = Properties.Resources.CapitalSign_eng;
            }
            else if (letter == 'Z')
            {
                pictBr.Image = Properties.Resources.Z_br;
                pictBr2.Image = Properties.Resources.CapitalSign_br;
                pictEng.Image = Properties.Resources.capZ_eng;
                pictEng2.Image = Properties.Resources.CapitalSign_eng;
            }
        }

        private void letterImagePunc(PictureBox pictBr, PictureBox pictEng, ListBox      ↙
    listName)
        {//Method used to load images into the chosen pictureBoxes.
         //Used in Learning Mode when Punctuation list is chosen.

            if (listName.SelectedIndex == 0)
            {
                pictBr.Image = Properties.Resources.Apostrophe_br;
                pictEng.Image = Properties.Resources.Apostrophe_eng;
            }
            else if (listName.SelectedIndex == 1)
            {
                pictBr.Image = Properties.Resources.Bracket_br;
                pictureBox29.Image = Properties.Resources.Bracket_br;
                pictEng.Image = Properties.Resources.BracketClose_eng;
                pictureBox25.Image = Properties.Resources.BracketOpen_eng;
            }
            else if (listName.SelectedIndex == 2)
            {
                pictBr.Image = Properties.Resources.Comma_br;
                pictEng.Image = Properties.Resources.Comma_eng;
            }
            else if (listName.SelectedIndex == 3)
        {
            pictBr.Image = Properties.Resources.Exclamation_br;
            pictEng.Image = Properties.Resources.Exclamation_eng;
        }
        else if (listName.SelectedIndex == 4)
        {
            pictBr.Image = Properties.Resources.Hyphen_br;
            pictEng.Image = Properties.Resources.Hyphen_eng;
        }
        else if (listName.SelectedIndex == 5)
        {
            pictBr.Image = Properties.Resources.Period_br;
            pictEng.Image = Properties.Resources.Period_eng;
        }
        else if (listName.SelectedIndex == 6)
        {
            pictBr.Image = Properties.Resources.QuoteOpen_br;
            pictEng.Image = Properties.Resources.QuestionMark_eng;
            //Quotation open and question mark are the same.
        }
        else if (listName.SelectedIndex == 7)
        {
            pictBr.Image = Properties.Resources.QuoteClose_br;
            pictureBox29.Image = Properties.Resources.QuoteOpen_br;
            pictEng.Image = Properties.Resources.QuoteClose_eng;
            pictureBox25.Image = Properties.Resources.QuoteOpen_eng;
        }
        else if (listName.SelectedIndex == 8)
        {
            pictBr.Image = Properties.Resources.Semicolon_br;
            pictEng.Image = Properties.Resources.Semicolon_eng;
        }
        else
        {
        }
    }

    private void numImage(char letter, PictureBox pictBr, PictureBox pictEng)
    {//Method used to load images into the chosen pictureBoxes.
     //Used in Learning Mode when "Numbers" list is chosen.

     //Deals with every digit except the first digit in the number.
     //   (see firstDigitImage method below)

        if (letter == '1')
        {
            pictBr.Image = Properties.Resources.A_br;
            pictEng.Image = Properties.Resources._1_eng;
        }
        else if (letter == '2')
        {
            pictBr.Image = Properties.Resources.B_br;
            pictEng.Image = Properties.Resources._2_eng;
        }
        else if (letter == '3')
        {
            pictBr.Image = Properties.Resources.C_br;
            pictEng.Image = Properties.Resources._3_eng;
        }
        else if (letter == '4')
        {
            pictBr.Image = Properties.Resources.D_br;
            pictEng.Image = Properties.Resources._4_eng;
        }
        else if (letter == '5')
        {
            pictBr.Image = Properties.Resources.E_br;
            pictEng.Image = Properties.Resources._5_eng;
        }
        else if (letter == '6')
        {
            pictBr.Image = Properties.Resources.F_br;
            pictEng.Image = Properties.Resources._6_eng;
        }
        else if (letter == '7')
        {
            pictBr.Image = Properties.Resources.G_br;
            pictEng.Image = Properties.Resources._7_eng;
        }
        else if (letter == '8')
        {
            pictBr.Image = Properties.Resources.H_br;
            pictEng.Image = Properties.Resources._8_eng;
        }
        else if (letter == '9')
        {
            pictBr.Image = Properties.Resources.I_br;
            pictEng.Image = Properties.Resources._9_eng;
        }
        else if (letter == '0')
        {
            pictBr.Image = Properties.Resources.J_br;
            pictEng.Image = Properties.Resources._0_eng;
        }
    }

    private void firstDigitImage(char letter, PictureBox pictBr, PictureBox pictBr2, ↙
PictureBox pictEng, PictureBox pictEng2)
    {//Method used to load images into the chosen pictureBoxes.
     //Used in Learning mode when "Numbers" list is chosen.

      //ONLY deals with first digit (left-most) in the number.
      //   (see numImage method above for all other digits.)

      //Reminder: Upper case notation requires two letters in Braille.

        if (letter == '1')
        {
            pictBr.Image = Properties.Resources.A_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._1_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '2')
        {
            pictBr.Image = Properties.Resources.B_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._2_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '3')
        {
            pictBr.Image = Properties.Resources.C_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._3_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '4')
        {
            pictBr.Image = Properties.Resources.D_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._4_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '5')
        {
            pictBr.Image = Properties.Resources.E_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._5_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '6')
        {
            pictBr.Image = Properties.Resources.F_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._6_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '7')
        {
            pictBr.Image = Properties.Resources.G_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._7_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '8')
        {
            pictBr.Image = Properties.Resources.H_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._8_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '9')
        {
            pictBr.Image = Properties.Resources.I_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._9_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
        else if (letter == '0')
        {
            pictBr.Image = Properties.Resources.J_br;
            pictBr2.Image = Properties.Resources.NumberSign_br;
            pictEng.Image = Properties.Resources._0_eng;
            pictEng2.Image = Properties.Resources.Numbers_eng;
        }
    }
    //Methods for updating images end here

    //**Methods for timers start here
    private void timer1_Tick_1(object sender, EventArgs e)
    {// Method used when the timer ticks down one second.
        // 1 second  = 1000ms.
        if (secTime > 0)
        {
            secTime--;
            if (secTime > 10)
            {
                label36.Text = Convert.ToString(secTime);
            }
            else if (secTime > 0 && secTime < 10)
            {
                label36.Text = Convert.ToString("0" + secTime);
            }
        }
        else
        {
            label36.Text = "00";
        }

        if (label35.Text == "10")
        {
            label35.Text = "9";
        }
        }

        private void timer1_Tick(object sender, EventArgs e)
        {//Timer used to count down minutes

            if (minTime > 0)
            {//Resets second timer to count down another minute
                minTime--;
                label35.Text = Convert.ToString(minTime);
                secTime = 60;
            }
            else
            {//Implies 10mins has passed.
                label36.Text = "00";
                label35.Text = "0";
                MessageBox.Show("TIME IS UP");

                panelTestDone();
                timerReset();
                statsTestRec();
                //Record stats. "TIME IS UP" event can only occur in Timed test mode.
            }
        }

        private void timerReset()
        {//Method used to disable/reset timers when exiting a menu.

            timerSec.Stop();
            timerMin.Stop();

            secTime = secDuration;
            minTime = minDuration;
            //Resets counter.
            label35.Text = "10";
            label36.Text = "00";
        }

        //Methods for timers end here


        //**Methods for dealing with GUI and panels start here
        private void panelHideAll()
        {//Method used to hide all Panels when changing between modes or
         //navigating the intial menu.

            panelNav.Hide();
            panelNav1.Hide();
            panelTest1.Hide();
            panelTest2.Hide();
            panelTest3.Hide();
            panelTest4.Hide();
            panelTest5.Hide();
            panelTest6.Hide();
            panelTest7.Hide();
            panelTestHidden.Hide();
            panelImage.Hide();
            panelList1.Hide();
            panelTestNav.Hide();
            panelHidden.Hide();
            panelNavSpeech.Hide();
            panelRandStats.Hide();
            panelTestCancel.Hide();
            panelCheat.Hide();
            panelCheat2.Hide();
            panelLearn.Hide();
        }

        private void panelSpeech()
```

```
{//Method used when going into Speech Recognition mode.

    textBox7.Clear();
    navButton = 1;

    //Creates MATLAB object to refer to, if MATLAB is
    //available.

    panelHideAll();

    panelImage.Show();
    panelImage.BringToFront();

    panelNav1.Show();
    panelNav1.BringToFront();

    panelNavSpeech.Location = new Point(90, 169);
    panelNavSpeech.Size = new Size(300, 308);
    panelNavSpeech.Show();
    panelNavSpeech.BringToFront();
    pictureBoxHEADER.Image = Properties.Resources.speechLabel;

    textBox7.Text = "-1";
    //Used to reset the guessed letters.

    //Code used to load list into Speech recog.
    //List is strictly set to alphabet only. **DO NOT ALLOW CHANGING LISTS.**
    string temp = "";
    int countDown = 0;

    //Prepares streamReader to read in the available alphabet letters.
    clearLists();
    FileInfo file = new FileInfo("Dictionary\\one.txt");
    StreamReader stRead = file.OpenText();

    while (!stRead.EndOfStream)
    {//Reads in alphabet one at a time.
        temp = Convert.ToString(stRead.ReadLine());
        listBoxONE.Items.Add(temp);
    }

    //Flushes streamReader and removes reference
    stRead.Close();
    textBoxONE.Text = listBoxONE.Items.Count.ToString();
    textBoxONECH.Text = "1";

    //No maximum threshold needed, all 26 letters to be added.
    countDown = listBoxONE.Items.Count;
    maxTemp = countDown;

    while (countDown > 0)
    {//Randomly adds each alphabet letter to the list, one at a time.
        Random objRan = new Random();
        int rand = objRan.Next(0, listBoxONE.Items.Count);
        temp = Convert.ToString(listBoxONE.Items[rand]);
        listBoxONE.Items.Remove(temp);
        listBoxTest.Items.Add(temp);
        countDown--;
    }

    //Defaults pictureBox setting to show the letter.
    //Unnecessary to guess for correction, since main feature here is
    //speech recognition capabilities.

    engWord = 1;
    listBoxTest.SelectedIndex = 0;
    loadTestImage();
```

```
            try
            {//Attempts to initiate MATLAB command window.
                ml = new MLApp.MLAppClass();
            }
            catch
            {//Error handler if MATLAB is not installed.
                MessageBox.Show("MATLAB not installed. Unable to run mode.");
                clearLists();
                panelHideAll();
                panelMainMenu();
                timerReset();
            }

            microSpeech();

        }

        private void panelTestNavigation()
        {//Method used to change from main menu to Test Navigation Menu

            pictureBoxHEADER.Image = Properties.Resources.testLabel;
            panelHideAll();
            panelTestNav.Show();
            panelTestNav.BringToFront();
            panelTestNav.Dock = DockStyle.Fill;
            // panelTestNav.Location = new Point(500, 250);

            navButton = 0;
        }

        private void panelTestTimed()
        {//Method used after choosing a Timed Test.
         //Displays the proper panels.

            panelTest2.Show();
            panelTest2.BringToFront();

            panelTest1.Show();
            panelTest1.BringToFront();

            panelTest6.Show();
            panelTest6.BringToFront();
            panelTest6.Location = new Point(550, 250);

            panelTest7.Hide();
            panelTest7.BringToFront();
            panelTest7.Location = new Point(49, 361);

            panelImage.Size = new Size(369, 102);
            panelImage.Show();

            // panelImage will remain hidden until pressing start.
            // panelTest3 will remain hidden until pressing start.

            statsFileRead();
            //Update stats incase returning to a test from the main menu.
        }

        private void panelTestRandom()
        {//Method used after choosing a Random Test.
         //Displays the proper panels.

            panelRandStats.Show();
            panelRandStats.BringToFront();
            panelRandStats.Size = new Size(227, 217);
            panelRandStats.Location = new Point(49, 124);
            panelTest5.Show();
            panelTest5.BringToFront();
            panelTest5.Location = new Point(49, 470);

            panelTest6.Show();
            panelTest6.BringToFront();
            panelTest6.Location = new Point(550, 250);

            panelTest7.Hide();
            panelTest7.BringToFront();
            panelTest7.Location = new Point(49, 361);

            panelImage.Size = new Size(369, 102);
            panelImage.Show();

            // panelImage will remain hidden until pressing start.
            // panelTest3 will remain hidden until pressing start.

            statsFileRead();
            //Update stats incase returning to a test from the main menu.
        }

        private void panelTestNavigationHideAll()
        {//Method used in Test Navigation Menu.
         //Hides timed/random buttons for each option except for the
         //one being pressed.

            button21.Hide();
            button22.Hide();
            // Buttons for "One Letter Word"

            button25.Hide();
            button24.Hide();
            // Buttons for "Two Letter Words"

            button27.Hide();
            button26.Hide();
            // Buttons for "Three Letter Words"

            button33.Hide();
            button28.Hide();
            // Buttons for "Four Letter Words"

            button35.Hide();
            button34.Hide();
            // Buttons for "All Words"

            button37.Hide();
            button36.Hide();
            // Buttons for "Numbers"
        }

        private void panelLearningMode()
        {//Method used when entering Learning Mode.
         //Hides unnecessary panels and shows needed ones.

            panelHideAll();
            panelList1.Show();
            panelList1.Size = new Size(358, 418);
            panelList1.Location = new Point(49, 150);
            panelList1.BringToFront();

            panelImage.Show();
            panelImage.BringToFront();

            panelLearn.Size = new Size(500, 500);
            panelLearn.Location = new Point(422, 100);
            panelLearn.Show();
            panelLearn.BringToFront();
            //Used to show instructions before starting.

            panelNav1.Show();
            panelNav1.BringToFront();

            button17.Hide();
            //Hide modify list option until mode is stared.

            pictureBoxHEADER.Image = Properties.Resources.learnLabel;

            panelCheat.BringToFront();
            panelCheat.Show();

            navButton = 0;
        }

        private void panelMainMenu()
        {//Method used to return to the Main (startup) menu.

            panelNav.Show();
            panelNav.BringToFront();
            panelNav.Dock = DockStyle.Fill;

            button51.Hide();
            button52.Hide();
            label72.Hide();
            //Hide reset stats buttons.

            button21.Hide();
            button22.Hide();
            button25.Hide();
            button24.Hide();
            button27.Hide();
            button26.Hide();
            button33.Hide();
            button28.Hide();
            button35.Hide();
            button34.Hide();
            button37.Hide();
            button36.Hide();
            //Hide buttons from Test Menu.

            imageBoxClear();
            //Clears out all image boxes.
        }

        private void imageBoxClear()
        {
            pictureBox24.Image = null;
            pictureBox25.Image = null;
            pictureBox26.Image = null;
            pictureBox27.Image = null;
            pictureBox28.Image = null;
            pictureBox29.Image = null;
            pictureBox30.Image = null;
            pictureBox31.Image = null;
        }
        //Methods for dealing with GUI and panels end here


        //**Methods used to handle keypresses start here
        private void textBox4_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13)
            {

                if (panelTest6.Visible)
```

```
                {
                    MessageBox.Show("Click START to begin.");
                    textBox4.Text = "";
                }
                else if (textBox4.Text == "")
                {
                    //Do Nothing.
                }
                else if (textBox4.Text.Length != testString.Length)
                {
                    textBox4.Text = "";
                    MessageBox.Show("Entering improper number of letters. Try again!");
                }
                else
                {
                    guessString = textBox4.Text;
                    textBox4.Text = "";
                    int i = 0;
                    int wordLength = testString.Length;
                    for (i = 0; i < wordLength; i++)
                    {// Compares to the correctly guessed word.

                        if (guessString[i] == testString[i])
                        {
                            indexString[i] = 1;
                        }
                    }
                    if (engWord == 1 || engWord == 2)
                    {
                        testWord();
                    }
                    if (guessString == testString)
                    {
                        testRight();
                    }
                    else
                    {
                        updateWrongTest();
                    }
                }
            }
        }

        private void button6_KeyPress(object sender, KeyPressEventArgs e)
        {

        }

        private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13)
            {
                if (panelTest6.Visible)
                {
                    MessageBox.Show("Click START to begin.");
                    textBox1.Text = "";
                }
                else if (textBox1.Text == "")
                {
                }
                else if (textBox1.Text.Length != testString.Length)
                {
                    textBox1.Text = "";
                    MessageBox.Show("Entering improper number of letters. Try again!");
                }
                else
                {
                    guessString = textBox1.Text;
                    textBox1.Text = "";
                    int i = 0;
                    int wordLength = testString.Length;
                    for (i = 0; i < wordLength; i++)
                    {// Compares to the correctly guessed word.

                        if (guessString[i] == testString[i])
                        {
                            indexString[i] = 1;
                        }
                    }
                    if (engWord == 1 || engWord == 2)
                    {
                        testWord();
                    }
                    if (guessString == testString)
                    {
                        testRight();
                    }
                    else
                    {
                        updateWrongRand();
                    }
                }
            }
        }
        //Methods used to handle keypresses end here


        //**Button control starts here. NO MORE METHODS.
        private void button1_Click(object sender, EventArgs e)
        {
            panelTestNavigation();
        }

        private void buttonDock_Click(object sender, EventArgs e)
        {
            panelNav.Dock = DockStyle.Fill;
        }

        private void buttonUndock_Click(object sender, EventArgs e)
        {
            panelNav.Dock = DockStyle.None;
        }

        private void button32_Click(object sender, EventArgs e)
        {
            panelLearningMode();
        }

        private void button38_Click(object sender, EventArgs e)
        {
            panelSpeech();
            matlabINPUT();
        }

        private void button39_Click(object sender, EventArgs e)
        {
            panelTest2.Hide();

            panelHidden.Hide();
            panelNav.Show();
            panelNav.Dock = DockStyle.Fill;
        }

        private void button17_Click(object sender, EventArgs e)
        {
            if (button17.Text.ToString() == "Modify Current List")
```

```
            {
                button17.Text = "Hide Options";
                textBox3.Show();
                button19.Show();
                button31.Show();
            }
            else
            {
                textBox3.Clear();
                button17.Text = "Modify Current List";
                textBox3.Hide();
                button19.Hide();
                button31.Hide();
            }
        }

        private void button31_Click(object sender, EventArgs e)
        {
            if (listBox2.Items.Contains(textBox3.Text))
            {
                MessageBox.Show("Error: Word already exists in list!"); // Do Nothing.
            }
            else
            {
                checkAdd();
            }
        }

        private void button19_Click(object sender, EventArgs e)
        {
            try
            { //

                if (listBox2.Items.Contains(textBox3.Text))
                {
                    checkRem();
                }
                else if(listBox2.Items.Contains( Convert.ToInt32(textBox3.Text) ))
                {
                    checkRem();
                }
                else
                {
                    MessageBox.Show("Error: Word not found in list!"); // Do Nothing.
                }
            }
            catch
            {
                if (listBox2.Items.Count == 0)
                {
                    MessageBox.Show("Error: List is empty!"); // Do Nothing.
                }
                else
                {
                    MessageBox.Show("Error: Word not found in list!"); // Do Nothing.
                }
            }
        }

        private void button29_Click(object sender, EventArgs e)
        {
            listPopup();
        }

        private void button30_Click(object sender, EventArgs e)
        {
            changeList();
        }
```

```
private void buttonNav2_Click(object sender, EventArgs e)
{
    panelNav.Hide();
    panelTest2.Hide();
    panelHidden.Show();
    panelHidden.Dock = DockStyle.Fill;
}

private void button18_Click(object sender, EventArgs e)
{
    if (listBox2.Items.Count == 0)
    {
        MessageBox.Show("No words in list to choose from!");
    }
    else
    {
        randWord();
        int val = 28;
        // Start at pictureBox28 for Braille letters.
        // Start at pictureBox24 for English letters.
        // Only need to reference pictureBox28 for method.
        chooseWord(val);
    }
}

private void button16_Click(object sender, EventArgs e)
{
    int val = 28;
    // Start at pictureBox28 for Braille letters.
    // Start at pictureBox24 for English letters.
    // Only need to reference pictureBox28 for method.
    microLearn();
    chooseWord(val);
}

private void button39_Click_1(object sender, EventArgs e)
{
    testTog2(listBox2);
}

private void button40_Click(object sender, EventArgs e)
{
    panelTestHidden.BringToFront();
    panelTestHidden.Show();

    panelHidden.BringToFront();
    panelHidden.Show();

    panelList1.BringToFront();
    panelList1.Show();
}

private void button42_Click(object sender, EventArgs e)
{
    if (listBoxTest.Items.Count != 0)
    {// Safety net to prevent starting without any words in list.
        panelTestCancel.Size = new Size(331, 91);
        panelTestCancel.Location = new Point(34, 5);
        panelTestCancel.Show();
        panelTestCancel.BringToFront();

        listBoxTest.SelectedIndex = 0;
        testString = listBoxTest.Text;
        testWord();

        panelTest6.SendToBack();
        panelTest6.Hide();
        panelImage.Show();
        panelImage.BringToFront();
        panelImage.Size = new Size(369, 518);

        panelTest3.Show();
        panelTest3.BringToFront();

        panelTest7.Show();

        panelCheat.BringToFront();
        panelCheat.Show();
        microTest();

        // Only show the panels panelImage and panelTest3 upon
        // pressing start. Hide otherwise.

        if (textBoxTestTimer.Text == "1")
        {

            timerSec.Start();
            timerMin.Start();

            label35.Text = "10";
            label36.Text = "00";
        }
    }
    else
    {
        panelHideAll();
        panelMainMenu();
        timerReset();
        MessageBox.Show("No words in list. Cannot run test!");
    }

}

private void button44_Click(object sender, EventArgs e)
{
    if (serialPort1.IsOpen)
    {
        string temp = textBox2.Text;
        serialPort1.Write(temp);
    }
    else {
    }
    //string temp = textBox2.Text;
    ////configuring the serial port
    //serialPort1.PortName = "COM3";
    //serialPort1.BaudRate = 9600;
    //serialPort1.DataBits = 8;
    //serialPort1.Parity = Parity.None;
    //serialPort1.StopBits = StopBits.One;

    ////opening the serial port
    //serialPort1.Open();

    ////write data to serial port
    //serialPort1.Write(temp);

    ////close the port
    //serialPort1.Close();

}

private void button45_Click(object sender, EventArgs e)
{
    int length;
```

```
            int num1, num2, num3;
            int var, total;
            char temp1, temp2, temp3;
            string str1, str2, str3;
            textBox6.AppendText(ml.Execute(textBox5.Text).Replace("\u000a".ToString(),    ↙
    Environment.NewLine));
            length = textBox6.Text.Length -1;
            temp1 = textBox6.Text[length - 6];
            temp2 = textBox6.Text[length - 5];
            temp3 = textBox6.Text[length - 4];
            textBox6.Clear();

            if (temp1 == '1')
            {
                textBox7.Text = "100";
            }
            else
            {
                str1 = Convert.ToString(temp1);
                str2 = Convert.ToString(temp2);
                str3 = Convert.ToString(temp3);

                int.TryParse(str1, out var);
                num1 = var*100;
                int.TryParse(str2, out var);
                num2 = var*10;
                int.TryParse(str3, out var);
                num3 = var;
                total = num2 + num3;
                textBox7.Text = total.ToString();
            }

        }

        private void button10_Click(object sender, EventArgs e)
        {//Code for clicking "One Letter Word" under Test Navigation
            panelTestNavigationHideAll();
            button21.Show();
            button22.Show();
        }

        private void button11_Click(object sender, EventArgs e)
        {//Code for clicking "Two Letter Words" under Test Navigation
            panelTestNavigationHideAll();
            button25.Show();
            button24.Show();
        }

        private void panelNav_Paint(object sender, PaintEventArgs e)
        {

        }

        private void button12_Click(object sender, EventArgs e)
        {//Code for clicking "Three Letter Words" under Test Navigation
            panelTestNavigationHideAll();
            button27.Show();
            button26.Show();
        }

        private void button13_Click(object sender, EventArgs e)
        {//Code for clicking "Four Letter Words" under Test Navigation
            panelTestNavigationHideAll();
            button33.Show();
            button28.Show();
        }

        private void button14_Click(object sender, EventArgs e)
```

```
{//Code for clicking "All Words" under Test Navigation
    panelTestNavigationHideAll();
    button35.Show();
    button34.Show();
}

private void button20_Click(object sender, EventArgs e)
{//Code for clicking "Four Letter Words" under Test Navigation
    panelTestNavigationHideAll();
    button37.Show();
    button36.Show();
}

private void button15_Click(object sender, EventArgs e)
{
    clearLists();
    panelHideAll();
    panelMainMenu();
    timerReset();

    navButton = 0;
    engWord = 0;
    listBoxTest.Items.Clear();
    //Reset engWord to 0 so that it defaults to hiding English Word.
    //Clear listBoxTest in case it is being called from Speech Rec. mode.

    listBox2.Items.Clear();
    listBox3.Items.Clear();
    button29.Text = "Choose Word Selection";
    listBox3.Hide();
    button30.Hide();
    //Clears all listboxes for Learning mode

    button55.Text = "Correctly Guessed";
    button56.Text = "Show All Letters";
    engWord = 0;
}

private void button46_Click(object sender, EventArgs e)
{
    panelHideAll();
    panelMainMenu();
}

private void button21_Click(object sender, EventArgs e)
{
    textBoxTestTimer.Text = "1";
    textBoxONECH.Text = "1";
    loadTestWords();
    panelHideAll();
    panelTestTimed();
}

private void button25_Click(object sender, EventArgs e)
{
    textBoxTestTimer.Text = "1";
    textBoxTWOCH.Text = "1";
    loadTestWords();
    panelHideAll();
    panelTestTimed();
}

private void button22_Click(object sender, EventArgs e)
{
    textBoxRandom.Text = "1";
    textBoxONECH.Text = "1";
    loadTestWords();
    panelHideAll();
```

```
            panelTestRandom();
        }

        private void button27_Click(object sender, EventArgs e)
        {
            textBoxTestTimer.Text = "1";
            textBoxTHREECH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestTimed();
        }

        private void button26_Click(object sender, EventArgs e)
        {
            textBoxRandom.Text = "1";
            textBoxTHREECH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestRandom();
        }

        private void button24_Click(object sender, EventArgs e)
        {
            textBoxRandom.Text = "1";
            textBoxTWOCH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestRandom();
        }

        private void button33_Click(object sender, EventArgs e)
        {
            textBoxTestTimer.Text = "1";
            textBoxFOURCH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestTimed();
        }

        private void button28_Click(object sender, EventArgs e)
        {
            textBoxRandom.Text = "1";
            textBoxFOURCH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestRandom();
        }

        private void button35_Click(object sender, EventArgs e)
        {
            textBoxTestTimer.Text = "1";
            textBoxALLCH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestTimed();
        }

        private void button34_Click(object sender, EventArgs e)
        {
            textBoxRandom.Text = "1";
            textBoxALLCH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestRandom();
        }

        private void button37_Click(object sender, EventArgs e)
        {
            textBoxTestTimer.Text = "1";
            textBoxNUMCH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestTimed();
        }

        private void button36_Click(object sender, EventArgs e)
        {
            textBoxRandom.Text = "1";
            textBoxNUMCH.Text = "1";
            loadTestWords();
            panelHideAll();
            panelTestRandom();
        }

        private void button23_Click(object sender, EventArgs e)
        {

            if (panelTest6.Visible)
            {
                MessageBox.Show("Click START to begin.");
                textBox4.Text = "";
            }
            else if (textBox4.Text == "")
            {
                //Do Nothing.
            }
            else if (textBox4.Text.Length != testString.Length)
            {
                textBox4.Text = "";
                MessageBox.Show("Entering improper number of letters. Try again!");
            }
            else
            {
                guessString = textBox4.Text;
                textBox4.Text = "";
                int i = 0;
                int wordLength = testString.Length;
                for (i = 0; i < wordLength; i++)
                {// Compares to the correctly guessed word.

                    if (guessString[i] == testString[i])
                    {
                        indexString[i] = 1;
                    }
                }
                if (engWord == 1 || engWord == 2)
                {
                    testWord();
                }
                if (guessString == testString)
                {
                    testRight();
                }
                else
                {
                    updateWrongTest();
                }

            }


        }

        private void button6_Click(object sender, EventArgs e)
        {
            if (panelTest6.Visible)
            {
                MessageBox.Show("Click START to begin.");
                textBox1.Text = "";
            }
            else if (textBox1.Text == "")
            {
            }
            else if (textBox1.Text.Length != testString.Length)
            {
                textBox1.Text = "";
                MessageBox.Show("Entering improper number of letters. Try again!");
            }
            else
            {
                guessString = textBox1.Text;
                textBox1.Text = "";
                int i = 0;
                int wordLength = testString.Length;
                for (i = 0; i < wordLength; i++)
                {// Compares to the correctly guessed word.

                    if (guessString[i] == testString[i])
                    {
                        indexString[i] = 1;
                    }
                }
                if (engWord == 1 || engWord == 2)
                {
                    testWord();
                }
                if (guessString == testString)
                {
                    testRight();
                }
                else
                {
                    updateWrongRand();
                }
            }

        }

        private void button7_Click(object sender, EventArgs e)
        {
            if (panelTest6.Visible)
            {
                MessageBox.Show("Test has not started!");
            }
            else
            {
                testWrong();
            }
        }

        private void button39_Click_2(object sender, EventArgs e)
        {
            int index = 0;
            string temp = "";
            if (listBoxRIGHT.SelectedIndex == -1)
            {// Nothing chosen in listBoxRIGHT. Analyzes listBoxWRONG.
                if (listBoxWRONG.SelectedIndex != -1)
                {// Converts the selected index in listboxWrong to a string.
                    // Finds the index in listboxTest and sends.
                    temp = listBoxWRONG.SelectedItem.ToString();
                    index = listBoxTest.Items.IndexOf(temp);
                    listBoxTest.SelectedIndex = index;
                    testWord();
                    microEndTest();
                }
                else
                {
                    MessageBox.Show("No word chosen!");
                }
            }
            else if (listBoxWRONG.SelectedIndex == -1)
            {// Nothing chosen in listBoxWRONG. Analyzes listBoxRIGHT.
                if (listBoxRIGHT.SelectedIndex != -1)
                {// Converts the selected index in listboxRIGHT to a string.
                    // Finds the index in listBoxTest and sends.
                    temp = listBoxRIGHT.SelectedItem.ToString();
                    index = listBoxTest.Items.IndexOf(temp);
                    listBoxTest.SelectedIndex = index;
                    testWord();
                    microEndTest();
                }
                else
                {
                    MessageBox.Show("No word chosen!");
                }
            }
        }

        private void button41_Click(object sender, EventArgs e)
        {
            clearLists();
            panelHideAll();
            panelMainMenu();
            timerReset();
        }

        private void panelHidden_Paint(object sender, PaintEventArgs e)
        {

        }

        private void button49_Click(object sender, EventArgs e)
        {
            statsCurrRand[0] = 0;
            statsCurrRand[1] = 0;
            statsCurrRand[2] = 0;
            statsCurrRand[3] = 0;
            statsRandUpdate();

            statsCurrTest[0] = 0;
            statsCurrTest[1] = 0;
            statsCurrTest[2] = 0;
            statsCurrTest[3] = 0;
            statsTestUpdate();

            timerReset();
            panelTest1.Hide();
            panelTest5.Hide();
            panelTest7.Hide();
            panelTestCancel.Hide();
            panelNav1.Show();
            panelNav1.BringToFront();

        }

        private void Form1_Load_1(object sender, EventArgs e)
        {

        }

        private void button50_Click(object sender, EventArgs e)
        {
            if (button51.Visible)
            {//Checks if one of the reset confirmation buttons are visible
                button51.Hide();
                button52.Hide();
                label72.Hide();
            }
            else
            {
                button51.Show();
                button52.Show();
                label72.Show();
            }
        }

        private void button52_Click(object sender, EventArgs e)
        {
            button51.Hide();
            button52.Hide();
            label72.Hide();
        }

        private void button51_Click(object sender, EventArgs e)
        {
            button51.Hide();
            button52.Hide();
            label72.Hide();

            statsArrTest[0] = 0;
            statsArrTest[1] = 0;
            statsArrTest[2] = 0;
            statsArrTest[3] = 0;

            statsArrRandom[0] = 0;
            statsArrRandom[1] = 0;
            statsArrRandom[2] = 0;
            statsArrRandom[3] = 0;

            statsTestRec();
            statsRandRec();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            panelTestNavigation();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            clearLists();
            panelLearningMode();
        }

        private void button43_Click(object sender, EventArgs e)
        {
            clearLists();
            panelHideAll();
            panelMainMenu();
            timerReset();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            panelSpeech();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            textBox7.Clear();
            soundByte = new SoundPlayer(Properties.Resources._310);
            soundByte.Play();

            int newIndex = 0;
            if (listBoxTest.SelectedIndex == 0)
            {//If at the start of the listbox, go to the bottom.
                newIndex = listBoxTest.Items.Count - 1;
                listBoxTest.SelectedIndex = newIndex;
            }
            else
            {//Move up the listBox.
                listBoxTest.SelectedIndex = listBoxTest.SelectedIndex - 1;
            }
            loadTestImage();
            matlabINPUT();

            textBox7.Text = "-1";

            microSpeech();
        }

        private void button9_Click(object sender, EventArgs e)
        {
            textBox7.Clear();
            soundByte = new SoundPlayer(Properties.Resources._310);
            soundByte.Play();
            if (listBoxTest.SelectedIndex == (listBoxTest.Items.Count -1))
            {//If at the bottom of the listbox, go to the top.
                listBoxTest.SelectedIndex = 0;
            }
            else
            {//Move down the listBox.
                listBoxTest.SelectedIndex = listBoxTest.SelectedIndex + 1;
            }
            loadTestImage();
            matlabINPUT();

            textBox7.Text = "-1";

            microSpeech();
        }

        private void button48_Click(object sender, EventArgs e)
        {
            string temp = "";
            soundByte = new SoundPlayer(Properties.Resources._380);
            soundByte.Play();
            Thread.Sleep(1700);
            temp = listBoxTest.SelectedItem.ToString();
            playSound(temp);
            soundByte.Play();
            //Used to play the actual answer.
        }

        private void button47_Click(object sender, EventArgs e)
        {
            soundByte = new SoundPlayer(Properties.Resources._390);
            soundByte.Play();
            Thread.Sleep(500);
            recordSpeech();
            soundByte = new SoundPlayer(Properties.Resources._400);
            soundByte.Play();
        }

        private void panelTestNav_Paint(object sender, PaintEventArgs e)
        {

        }
```

```
private void listBoxRIGHT_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)13)
    {
        int index = 0;
        string temp = "";
        if (listBoxRIGHT.SelectedIndex == -1)
        {// Nothing chosen in listBoxRIGHT. Analyzes listBoxWRONG.
            if (listBoxWRONG.SelectedIndex != -1)
            {// Converts the selected index in listboxWrong to a string.
                // Finds the index in listboxTest and sends.
                temp = listBoxWRONG.SelectedItem.ToString();
                index = listBoxTest.Items.IndexOf(temp);
                listBoxTest.SelectedIndex = index;
                testWord();
            }
            else
            {
                MessageBox.Show("No word chosen!");
            }
        }
        else if (listBoxWRONG.SelectedIndex == -1)
        {// Nothing chosen in listBoxWRONG. Analyzes listBoxRIGHT.
            if (listBoxRIGHT.SelectedIndex != -1)
            {// Converts the selected index in listboxRIGHT to a string.
                // Finds the index in listBoxTest and sends.
                temp = listBoxRIGHT.SelectedItem.ToString();
                index = listBoxTest.Items.IndexOf(temp);
                listBoxTest.SelectedIndex = index;
                testWord();
            }
            else
            {
                MessageBox.Show("No word chosen!");
            }
        }
    }
}

private void listBoxWRONG_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)13)
    {
        int index = 0;
        string temp = "";
        if (listBoxRIGHT.SelectedIndex == -1)
        {// Nothing chosen in listBoxRIGHT. Analyzes listBoxWRONG.
            if (listBoxWRONG.SelectedIndex != -1)
            {// Converts the selected index in listboxWrong to a string.
                // Finds the index in listboxTest and sends.
                temp = listBoxWRONG.SelectedItem.ToString();
                index = listBoxTest.Items.IndexOf(temp);
                listBoxTest.SelectedIndex = index;
                testWord();
            }
            else
            {
                MessageBox.Show("No word chosen!");
            }
        }
        else if (listBoxWRONG.SelectedIndex == -1)
        {// Nothing chosen in listBoxWRONG. Analyzes listBoxRIGHT.
            if (listBoxRIGHT.SelectedIndex != -1)
            {// Converts the selected index in listboxRIGHT to a string.
                // Finds the index in listBoxTest and sends.
                temp = listBoxRIGHT.SelectedItem.ToString();
                index = listBoxTest.Items.IndexOf(temp);
```

```
                    listBoxTest.SelectedIndex = index;
                    testWord();
                }
                else
                {
                    MessageBox.Show("No word chosen!");
                }
            }
        }
    }

    private void listBox2_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)13)
        {
            int val = 28;
            // Start at pictureBox28 for Braille letters.
            // Start at pictureBox24 for English letters.
            // Only need to reference pictureBox28 for method.
            chooseWord(val);
        }
    }

    private void listBox2_DoubleClick(object sender, EventArgs e)
    {
        int val = 28;
        // Start at pictureBox28 for Braille letters.
        // Start at pictureBox24 for English letters.
        // Only need to reference pictureBox28 for method.
        chooseWord(val);
    }

    private void listBox3_DoubleClick(object sender, EventArgs e)
    {
        changeList();
    }

    private void listBox3_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)13)
        {
            changeList();
        }
    }

    private void listBoxWRONG_DoubleClick(object sender, EventArgs e)
    {
        int index = 0;
        string temp = "";
        if (listBoxRIGHT.SelectedIndex == -1)
        {// Nothing chosen in listBoxRIGHT. Analyzes listBoxWRONG.
            if (listBoxWRONG.SelectedIndex != -1)
            {// Converts the selected index in listboxWrong to a string.
                // Finds the index in listboxTest and sends.
                temp = listBoxWRONG.SelectedItem.ToString();
                index = listBoxTest.Items.IndexOf(temp);
                listBoxTest.SelectedIndex = index;
                testWord();
            }
            else
            {
                MessageBox.Show("No word chosen!");
            }
        }
        else if (listBoxWRONG.SelectedIndex == -1)
        {// Nothing chosen in listBoxWRONG. Analyzes listBoxRIGHT.
            if (listBoxRIGHT.SelectedIndex != -1)
            {// Converts the selected index in listboxRIGHT to a string.
```

```
                // Finds the index in listBoxTest and sends.
                temp = listBoxRIGHT.SelectedItem.ToString();
                index = listBoxTest.Items.IndexOf(temp);
                listBoxTest.SelectedIndex = index;
                testWord();
            }
            else
            {
                MessageBox.Show("No word chosen!");
            }
        }
    }

    private void listBoxRIGHT_DoubleClick(object sender, EventArgs e)
    {
        int index = 0;
        string temp = "";
        if (listBoxRIGHT.SelectedIndex == -1)
        {// Nothing chosen in listBoxRIGHT. Analyzes listBoxWRONG.
            if (listBoxWRONG.SelectedIndex != -1)
            {// Converts the selected index in listboxWrong to a string.
                // Finds the index in listboxTest and sends.
                temp = listBoxWRONG.SelectedItem.ToString();
                index = listBoxTest.Items.IndexOf(temp);
                listBoxTest.SelectedIndex = index;
                testWord();
            }
            else
            {
                MessageBox.Show("No word chosen!");
            }
        }
        else if (listBoxWRONG.SelectedIndex == -1)
        {// Nothing chosen in listBoxWRONG. Analyzes listBoxRIGHT.
            if (listBoxRIGHT.SelectedIndex != -1)
            {// Converts the selected index in listboxRIGHT to a string.
                // Finds the index in listBoxTest and sends.
                temp = listBoxRIGHT.SelectedItem.ToString();
                index = listBoxTest.Items.IndexOf(temp);
                listBoxTest.SelectedIndex = index;
                testWord();
            }
            else
            {
                MessageBox.Show("No word chosen!");
            }
        }
    }

    private void numericList()
    {
        int limit = listBox2.Items.Count;
        int i, j, increment, temp;
        increment = 3;
        for (i = 0; i<limit; i++)
        {
            arr[i] = Convert.ToInt32(listBox2.Items[i]);
        }

        while (increment > 0)
        {
            for (i = 0; i < limit; i++)
            {
                j = i;
                temp = arr[i];

                while ((j >= increment) && (arr[j - increment] > temp))
                {
                    arr[j] = arr[j - increment];
                    j = j - increment;
                }

                arr[j] = temp;
            }

            if (increment / 2 != 0)
            {
                increment = increment / 2;
            }
            else if (increment == 1)
            {
                increment = 0;
            }
            else
            {
                increment = 1;
            }
        }

        listBox2.Items.Clear();

        for (i = 0; i < limit; i++)
        {
            listBox2.Items.Add(arr[i]);
        }
    }

    private void numericRight()
    {//Used for testing numbers.
        if (listBoxRIGHT.Items.Count != 0)
        {
            int limit = listBoxRIGHT.Items.Count;
            int i, j, increment, temp;
            increment = 3;
            for (i = 0; i < limit; i++)
            {
                arr[i] = Convert.ToInt32(listBoxRIGHT.Items[i]);
            }

            while (increment > 0)
            {//Shell sorting, to sort the list numerically
                for (i = 0; i < limit; i++)
                {
                    j = i;
                    temp = arr[i];

                    while ((j >= increment) && (arr[j - increment] > temp))
                    {
                        arr[j] = arr[j - increment];
                        j = j - increment;
                    }

                    arr[j] = temp;
                }

                if (increment / 2 != 0)
                {
                    increment = increment / 2;
                }
                else if (increment == 1)
                {
                    increment = 0;
                }
                else
                {
                    increment = 1;
                }
            }

            listBoxRIGHT.Items.Clear();

            for (i = 0; i < limit; i++)
            {//Add items to list now that they are sorted.
                listBoxRIGHT.Items.Add(arr[i]);
            }
        }
    }

    private void numericWrong()
    {// Used for testing numbers.
        if (listBoxWRONG.Items.Count != 0)
        {
            int limit = listBoxWRONG.Items.Count;
            int i, j, increment, temp;
            increment = 3;
            for (i = 0; i < limit; i++)
            {
                arr[i] = Convert.ToInt32(listBoxWRONG.Items[i]);
            }

            while (increment > 0)
            {//Shell sorting, to sort the list numerically
                for (i = 0; i < limit; i++)
                {
                    j = i;
                    temp = arr[i];

                    while ((j >= increment) && (arr[j - increment] > temp))
                    {
                        arr[j] = arr[j - increment];
                        j = j - increment;
                    }

                    arr[j] = temp;
                }

                if (increment / 2 != 0)
                {
                    increment = increment / 2;
                }
                else if (increment == 1)
                {
                    increment = 0;
                }
                else
                {
                    increment = 1;
                }
            }

            listBoxWRONG.Items.Clear();

            for (i = 0; i < limit; i++)
            {//Add items to list now that they are sorted.
                listBoxWRONG.Items.Add(arr[i]);
            }
        }
    }

    private void button55_Click(object sender, EventArgs e)
    {
        if (button55.Text == "Correctly Guessed")
        {
            button55.Text = "Hide Word";
            button56.Text = "Show All Letters";
            engWord = 2;
        }
        else
        {
            button55.Text = "Correctly Guessed";
            button56.Text = "Show All Letters";
            engWord = 0;
        }
        testWord();
    }

    private void button56_Click(object sender, EventArgs e)
    {
        if (button56.Text == "Show All Letters")
        {
            button55.Text = "Correctly Guessed";
            button56.Text = "Hide Word";
            engWord = 1;
        }
        else
        {
            button55.Text = "Correctly Guessed";
            button56.Text = "Show All Letters";
            engWord = 0;
        }
        testWord();
    }

    private void updateWrongTest()
    {
        statsCurrTest[2]++;
        statsCurrTest[3]++;
        statsTestUpdate();
    }

    private void updateWrongRand()
    {
        statsCurrRand[2]++;
        statsCurrRand[3]++;
        statsRandUpdate();
    }

    private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)13)
        {
            // Do nothing. Should NOT handle ENTER/RETURN key presses.
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        panelCheat2.BringToFront();
        panelCheat2.Show();
        panelCheat2.Dock = DockStyle.Fill;
    }

    private void button57_Click(object sender, EventArgs e)
    {
        if(button57.Text == "Numbers")
        {
            pictureBoxCheat.Image = Properties.Resources.cheatSheet2;
            button58.Text = "Alphabet";
            button57.Text = "Punctuation";
        }
        else if(button57.Text == "Punctuation")
        {
            pictureBoxCheat.Image = Properties.Resources.cheatSheet3;
            button58.Text = "Numbers";
            button57.Text = "Alphabet";
        }
        else
        {
            pictureBoxCheat.Image = Properties.Resources.cheatSheet1;
            button58.Text = "Punctuation";
            button57.Text = "Numbers";
        }

    }

    private void button58_Click(object sender, EventArgs e)
    {
        if (button58.Text == "Numbers")
        {
            pictureBoxCheat.Image = Properties.Resources.cheatSheet2;
            button58.Text = "Alphabet";
            button57.Text = "Puncutation";
        }
        else if (button58.Text == "Punctuation")
        {
            pictureBoxCheat.Image = Properties.Resources.cheatSheet3;
            button58.Text = "Numbers";
            button57.Text = "Alphabet";
        }
        else
        {
            pictureBoxCheat.Image = Properties.Resources.cheatSheet1;
            button58.Text = "Punctuation";
            button57.Text = "Numbers";
        }

    }

    private void button59_Click(object sender, EventArgs e)
    {
        panelCheat2.Hide();
    }

    private void button60_Click(object sender, EventArgs e)
    {
        panelImage.BringToFront();
        button17.Show();
    }

    private void button54_Click(object sender, EventArgs e)
    {
        speechVerify();
    }

    private void speechVerify()
    {//Tells user if their recorded answer is correct.
     //Before calling this function, MATLAB has already put
     //answer into C# program.
        string temp = "";
        temp = listBoxTest.SelectedItem.ToString();
        int val = -1;
        if (temp != null)
        {
            int.TryParse(textBox7.Text, out val);
        }

        if(val == 100)
        {//Means answer is correct.
            soundByte = new SoundPlayer(Properties.Resources._340);
            soundByte.Play();
            Thread.Sleep(1800);
            playSound(temp);
            soundByte.Play();
            Thread.Sleep(1000);
            //Reminder. playSound() only sets the file. Still needs to be played.

            soundByte = new SoundPlayer(Properties.Resources._310);
            soundByte.Play();
            if (listBoxTest.SelectedIndex == (listBoxTest.Items.Count - 1))
            {//If at the bottom of the listbox, go to the top.
                listBoxTest.SelectedIndex = 0;
            }
            else
            {//Move down the listBox.
                listBoxTest.SelectedIndex = listBoxTest.SelectedIndex + 1;
            }
            loadTestImage();
            matlabINPUT();

            textBox7.Text = "-1";
        }
        else if (val == -1)
        {//Means no answer recorded
            soundByte = new SoundPlayer(Properties.Resources._360);
            soundByte.Play();
        }
        else if (val == 50)
        {//Means answer cannot be interpreted.
            soundByte = new SoundPlayer(Properties.Resources._370);
            soundByte.Play();
            Thread.Sleep(2000);
            soundByte = new SoundPlayer(Properties.Resources._320);
            soundByte.Play();
        }
        else if (val > 0 && val <27)
        {//Means answer is incorrect.
            temp = textBox7.Text;
            soundByte = new SoundPlayer(Properties.Resources._350);
            soundByte.Play();
            playWrong(temp);
            Thread.Sleep(2200);
            if (soundByte != null)
            {
                soundByte.Play();
            }
        }
    }

    private void playWrong(string soundFile)
    {//Method used to play sounds.
     //Simply takes in a number and matches it to
     //respective letter value.
        if (soundFile == "1")
        {
            soundByte = new SoundPlayer(Properties.Resources._10);
        }
        else if (soundFile == "2")
        {
            soundByte = new SoundPlayer(Properties.Resources._20);
        }
        else if (soundFile == "3")
        {
            soundByte = new SoundPlayer(Properties.Resources._30);
        }
        else if (soundFile == "4")
        {
            soundByte = new SoundPlayer(Properties.Resources._40);
        }
```

```
else if (soundFile == "5")
{
    soundByte = new SoundPlayer(Properties.Resources._50);
}
else if (soundFile == "6")
{
    soundByte = new SoundPlayer(Properties.Resources._60);
}
else if (soundFile == "7")
{
    soundByte = new SoundPlayer(Properties.Resources._70);
}
else if (soundFile == "8")
{
    soundByte = new SoundPlayer(Properties.Resources._80);
}
else if (soundFile == "9")
{
    soundByte = new SoundPlayer(Properties.Resources._90);
}
else if (soundFile == "10")
{
    soundByte = new SoundPlayer(Properties.Resources._100);
}
else if (soundFile == "11")
{
    soundByte = new SoundPlayer(Properties.Resources._110);
}
else if (soundFile == "12")
{
    soundByte = new SoundPlayer(Properties.Resources._120);
}
else if (soundFile == "13")
{
    soundByte = new SoundPlayer(Properties.Resources._130);
}
else if (soundFile == "14")
{
    soundByte = new SoundPlayer(Properties.Resources._140);
}
else if (soundFile == "15")
{
    soundByte = new SoundPlayer(Properties.Resources._150);
}
else if (soundFile == "16")
{
    soundByte = new SoundPlayer(Properties.Resources._160);
}
else if (soundFile == "17")
{
    soundByte = new SoundPlayer(Properties.Resources._170);
}
else if (soundFile == "18")
{
    soundByte = new SoundPlayer(Properties.Resources._180);
}
else if (soundFile == "19")
{
    soundByte = new SoundPlayer(Properties.Resources._190);
}
else if (soundFile == "20")
{
    soundByte = new SoundPlayer(Properties.Resources._200);
}
else if (soundFile == "21")
{
    soundByte = new SoundPlayer(Properties.Resources._210);
}
```

```
        else if (soundFile == "22")
        {
            soundByte = new SoundPlayer(Properties.Resources._220);
        }
        else if (soundFile == "23")
        {
            soundByte = new SoundPlayer(Properties.Resources._230);
        }
        else if (soundFile == "24")
        {
            soundByte = new SoundPlayer(Properties.Resources._240);
        }
        else if (soundFile == "25")
        {
            soundByte = new SoundPlayer(Properties.Resources._250);
        }
        else if (soundFile == "26")
        {
            soundByte = new SoundPlayer(Properties.Resources._260);
        }
    }

    private void recordSpeech()
    {//Communicates with MATLAB to record user's speech
        int length;
        int num1, num2, num3;
        int var, total;
        char temp1, temp2, temp3;
        string str1, str2, str3;
        string temp = "";

        temp = letterNumber();
        //String that is sent to MATLAB. Used to tell MATLAB what letter is currently↙
 used.

        //Plays a beep noise indicating to speak into microphone.
        //Beep is slightly delayed to account for callign the matlab function.

        textBox5.Clear();
        textBox5.Text = "speechRec("+temp+")";
        textBox6.AppendText(ml.Execute(textBox5.Text).Replace("\u000a".ToString(),   ↙
Environment.NewLine));
        //Calls function in matlab. Text is to be placed in textbox6.

        length = textBox6.Text.Length - 1;
        temp1 = textBox6.Text[length - 6];
        temp2 = textBox6.Text[length - 5];
        temp3 = textBox6.Text[length - 4];
        textBox6.Clear();
        //Checks text and determines values MATLAB returned to C#

        if (temp1 == '1')
        {//Implies answer is correct.
            textBox7.Text = "100";
        }
        else
        {//Implies answer is incorrect. Determine specific output value.
            str1 = Convert.ToString(temp1);
            str2 = Convert.ToString(temp2);
            str3 = Convert.ToString(temp3);

            int.TryParse(str1, out var);
            num1 = var * 100;
            int.TryParse(str2, out var);
            num2 = var * 10;
            int.TryParse(str3, out var);
            num3 = var;
            total = num2 + num3;
            textBox7.Text = total.ToString();
        }
    }

    private void microLearn()
    {//Function used in Learning Application to command
     //microcontroller.
        try
        {
            string temp;
            string word;
            int length;
            if (textBoxCAPCH.Text == "1")
            {
                temp = "1";
            }
            else if (textBoxNUMCH.Text == "1")
            {
                temp = "2";
            }
            else if (textBoxPUNCH.Text == "1")
            {
                temp = "3";
            }
            else
            {
                temp = "0";
            }


            if (temp != "3")
            {//Add length and actual word to string.
                word = listBox2.SelectedItem.ToString();
                length = word.Length;
                temp += length.ToString();
                temp += listBox2.SelectedItem.ToString();
            }
            else if (temp == "3")
            {//Length dependent on punctuation.
                word = listBox2.SelectedItem.ToString();
                if (word == "Apostrophe")
                {
                    temp += "1";
                    temp += "a";
                }
                else if (word == "Brackets")
                {// Sends two brackets
                    temp += "2";
                    temp += "bc";
                }
                else if (word == "Comma")
                {
                    temp += "1";
                    temp += "d";
                }
                else if (word == "Exclamation")
                {
                    temp += "1";
                    temp += "e";
                }
                else if (word == "Hyphen")
                {
                    temp += "1";
                    temp += "f";
                }
                else if (word == "Period")
                {
                    temp += "1";
                    temp += "g";
                }
                else if (word == "Question Mark")
                {
                    temp += "1";
                    temp += "h";
                }
                else if (word == "Quotations")
                {
                    temp += "2";
                    temp += "ij";
                }
                else if (word == "Semicolon")
                {
                    temp += "1";
                    temp += "k";
                }
            }
            serialPort1.Write(temp);
        }
        catch
        {
        }

    }

    private void microTest()
    {//Function used in Testing Application to command microcontroller.
        try
        {
            string temp;
            string word;
            int length;
            if (textBoxCAPCH.Text == "1")
            {//Check if capital letters are being used.
                temp = "1";
            }
            else if (textBoxNUMCH.Text == "1")
            {//Check if numbers are being used.
                temp = "2";
            }
            else
            {
                temp = "0";
            }

            word = listBoxTest.SelectedItem.ToString();
            length = word.Length;
            temp += length.ToString();
            temp += listBoxTest.SelectedItem.ToString();

            serialPort1.Write(temp);
            //Send to microcontroller.
        }
        catch
        {
        }
    }

    private void microSpeech()
    {//Function used in Speech Rec. application to command microcontroller.
        try
        {//Sent word will always be one letter long.
            string temp = "01";
            string word;
            word = listBoxTest.SelectedItem.ToString();
            temp += word;
            serialPort1.Write(temp);
        }
        catch
        {
        }
    }

    private void microEndTest()
    {//Function used in Testing Application. Specifically for words at end of test.
        try
        {
            string temp;
            string word;
            int length;
            if (textBoxCAPCH.Text == "1")
            {//Check if capital letters are used.
                temp = "1";
            }
            else if (textBoxNUMCH.Text == "1")
            {//Check if numbers are used.
                temp = "2";
            }
            else
            {
                temp = "0";
            }

            if (listBoxRIGHT.SelectedIndex != -1)
            {//Check which of the two listBoxes has a selected index.
                //Runs if listboxRIGHT has a selected index.
                word = listBoxRIGHT.SelectedItem.ToString();
                length = word.Length;
                temp += length.ToString();
                temp += listBoxRIGHT.SelectedItem.ToString();

                serialPort1.Write(temp);
            }
            else if (listBoxWRONG.SelectedIndex != -1)
            {//Runs if listboxWrong has a selected index.
                word = listBoxWRONG.SelectedItem.ToString();
                length = word.Length;
                temp += length.ToString();
                temp += listBoxWRONG.SelectedItem.ToString();

                serialPort1.Write(temp);
            }
        }
        catch
        {
        }
    }

    private string letterNumber()
    {//Code used when sending numbers to MATLAB function.
     //Converts the letter to a respective integer value
        string temp = "";
        if(listBoxTest.SelectedItem.ToString() == "a")
        {
            temp = "01";
        }
        else if(listBoxTest.SelectedItem.ToString() == "b")
        {
            temp = "02";
        }
        else if(listBoxTest.SelectedItem.ToString() == "c")
        {
            temp = "03";
        }
```

```
else if(listBoxTest.SelectedItem.ToString() == "d")
{
    temp = "04";
}
else if(listBoxTest.SelectedItem.ToString() == "e")
{
    temp = "05";
}
else if(listBoxTest.SelectedItem.ToString() == "f")
{
    temp = "06";
}
else if(listBoxTest.SelectedItem.ToString() == "g")
{
    temp = "07";
}
else if(listBoxTest.SelectedItem.ToString() == "h")
{
    temp = "08";
}
else if(listBoxTest.SelectedItem.ToString() == "i")
{
    temp = "09";
}
else if(listBoxTest.SelectedItem.ToString() == "j")
{
    temp = "10";
}
else if(listBoxTest.SelectedItem.ToString() == "k")
{
    temp = "11";
}
else if(listBoxTest.SelectedItem.ToString() == "l")
{
    temp = "12";
}
else if(listBoxTest.SelectedItem.ToString() == "m")
{
    temp = "13";
}
else if(listBoxTest.SelectedItem.ToString() == "n")
{
    temp = "14";
}
else if(listBoxTest.SelectedItem.ToString() == "o")
{
    temp = "15";
}
else if(listBoxTest.SelectedItem.ToString() == "p")
{
    temp = "16";
}
else if(listBoxTest.SelectedItem.ToString() == "q")
{
    temp = "17";
}
else if(listBoxTest.SelectedItem.ToString() == "r")
{
    temp = "18";
}
else if(listBoxTest.SelectedItem.ToString() == "s")
{
    temp = "19";
}
else if(listBoxTest.SelectedItem.ToString() == "t")
{
    temp = "20";
}
```

```
            else if(listBoxTest.SelectedItem.ToString() == "u")
            {
                temp = "21";
            }
            else if(listBoxTest.SelectedItem.ToString() == "v")
            {
                temp = "22";
            }
            else if(listBoxTest.SelectedItem.ToString() == "w")
            {
                temp = "23";
            }
            else if(listBoxTest.SelectedItem.ToString() == "x")
            {
                temp = "24";
            }
            else if(listBoxTest.SelectedItem.ToString() == "y")
            {
                temp = "25";
            }
            else if(listBoxTest.SelectedItem.ToString() == "z")
            {
                temp = "26";
            }

            return temp;
        }
    }
}
```