INGENIERÍA EN SOFTWARE

LAB 03 – HASHTABLE

Objetivo:

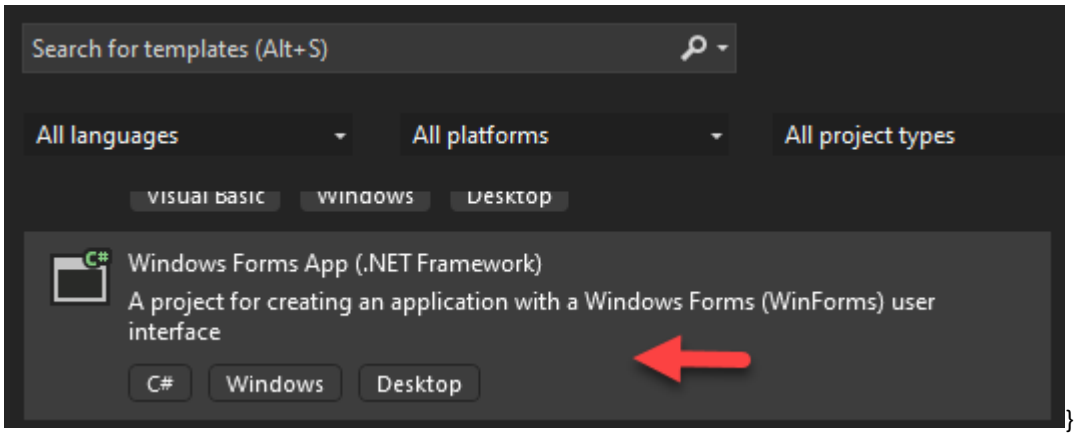
El objetivo de este laboratorio es

- Aprender a crear funciones en C# que realicen usen un hashtable para cargar información, revisar su rendimiento al cargar esta información y luego su búsqueda en el mismo.
- Partimos de un archivo llamado “EnglishWords.txt” el cual tiene una lista de palabras que la usaremos como nuestro diccionario
- Usaremos otro archivo llamado “” que será el archivo sobre el cual validaremos nuestro diccionario.
- Crear una función Hashtable y testear el tiempo que toma en poblar esta información, lo mismo con un diccionario y con un diccionario ordenado
- Aprender un poco más de diseño y programación de aplicaciones de escritorio con C#

1. Creación de proyecto

Creamos una carpeta vacía llamada “Lab04-Hastable”

Vamos a crear una nueva de escritorio, utilizando el siguiente template:



}

Como nombre de proyecto : “WinAppHastable”

Nos familiarizamos un poco con las opciones del IDE

2. Configuraciones iniciales del proyecto

En el archivo de configuración podemos colocar la siguiente configuración con el fín de no tener en hardcode los nombres de los archivos.

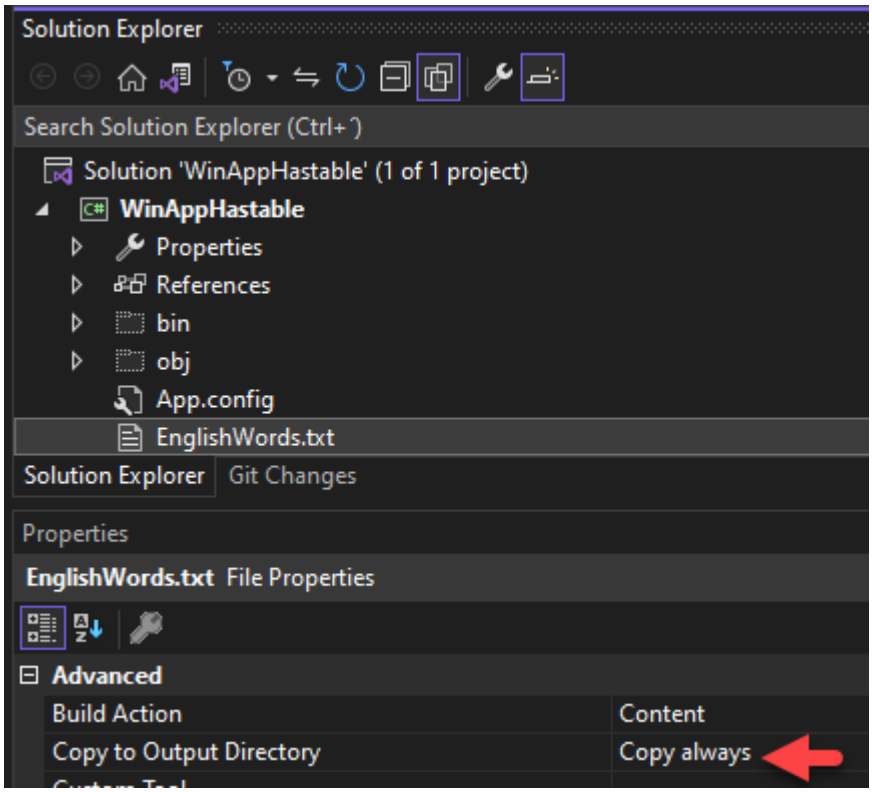
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
      <appSettings>
            <add key="dictionaryFile" value="EnglishWords.txt"/>
            <add key="targetFile" value="Roar.txt"/>
            <add key="missedWordsFile" value="MissedWordFile.txt"/>
      </appSettings>
```

```
    <startup>
        <supportedRuntime version="v4.0"
sku=".NETFramework,Version=v4.7.2" />
    </startup>
</configuration>
```

En la pantalla inicial “Form1” colocamos el siguiente código

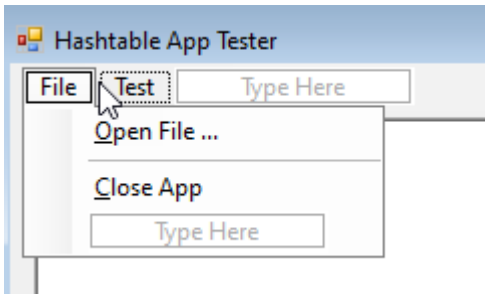
```
public partial class Form1 : Form
    {
        string filePath = string.Empty;
        string dictionaryFile = string.Empty;
        string targetFile = string.Empty;
        string missedWordsFile = string.Empty;
        public Form1()
        {
            dictionaryFile =
ConfigurationManager.AppSettings["dictionaryFile"];
            targetFile = ConfigurationManager.AppSettings["targetFile"];
            missedWordsFile =
ConfigurationManager.AppSettings["missedWordsFile"];
            InitializeComponent();
        }
```

Cargamos adicionalmente a nuestro proyecto los archivos EnglishWords.txt y Roar.txt, y en las propiedades de ambos archivos indicamos que se copien a la carpeta de salida

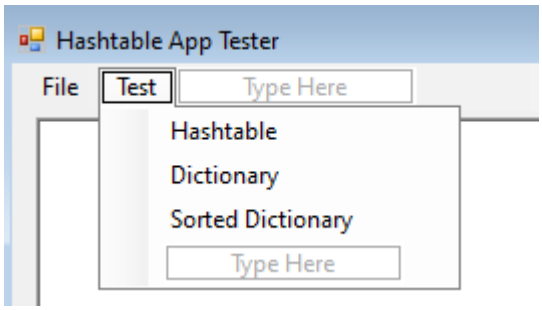


Diseñamos la pantalla de nuestra aplicación colocándole un título adecuado

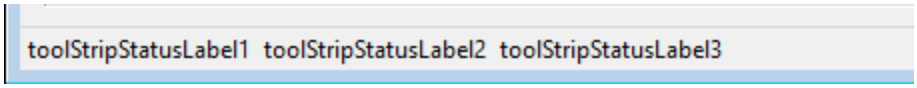
Creamos una opción de menú similar a:



Para testear los algoritmos realizamos igualmente este otro menú.



Añadimos también una barra de estado así:



3. Codificación de la opción Hastable

En esta opción podemos utilizar el siguiente código sugerido:

```
//Hastable object declaration
            Hashtable englishHashDictionary = new Hashtable();
            // Read the file and add words to hashtable
            DateTime startDate = DateTime.Now;
            foreach (string line in
System.IO.File.ReadLines(dictionaryFile))
            {
                if (!englishHashDictionary.ContainsKey(line))
                {
                    englishHashDictionary.Add(line, line);
                }
            }
            TimeSpan timeDiff = DateTime.Now - startDate;
            string result = string.Format("{0} add to Hashtable in {1}
ms", dictionaryFile, timeDiff.TotalMilliseconds);
            Console.WriteLine(result);
            toolStripStatusLabel1.Text = result;
            //Read target file to spell-check and put not fouded words in
a list
            IList<string> missedWords = new List<string>();
            foreach (string line in System.IO.File.ReadLines(targetFile))
            {
                string[] words = line.Replace(",", "").Replace("!",
"").Split(' ');
                foreach (string word in words)
                {
```

```
                    if (!englishHashDictionary.ContainsKey(word))
                    {
                        if (!missedWords.Contains(word))
                        {
                            missedWords.Add(word);
                        }
                    }
                }
            }
            if (missedWords.Count > 0)
            {
                MessageBox.Show(string.Join(Environment.NewLine,
missedWords));
                System.IO.File.WriteAllLines(missedWordsFile,
missedWords);
            }
            Console.WriteLine("File processed!!");
```

4. Codificación de la opción Dictionary

Basándonos en la opción anterior colocar el código necesario para usar un diccionario:

La declaración del diccionario puede ser:

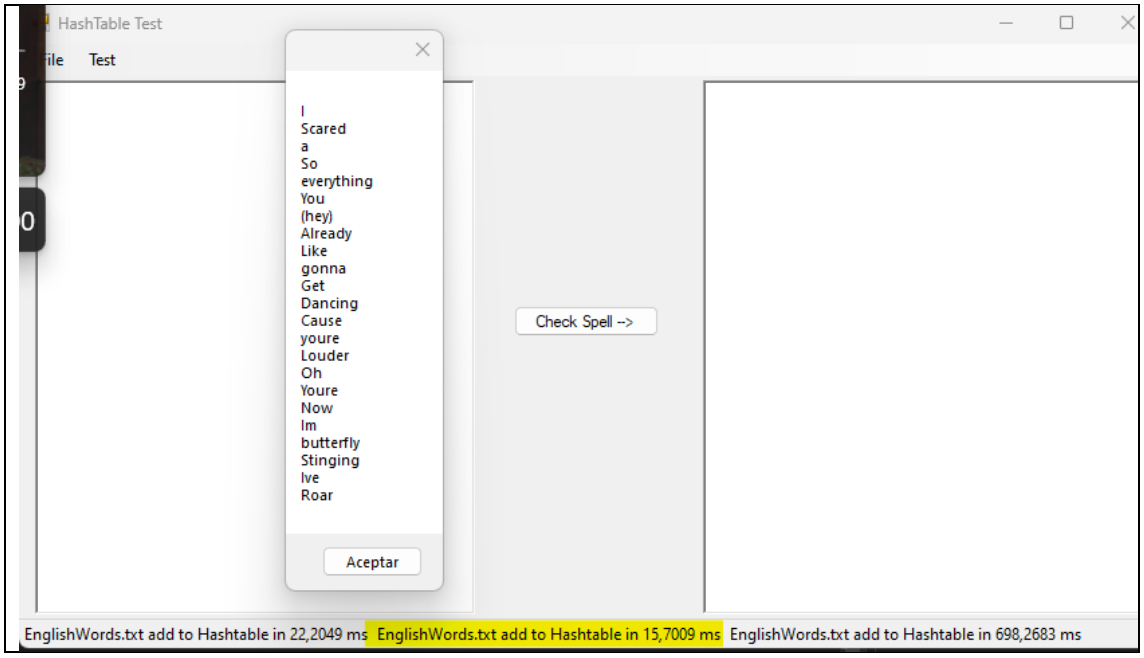
```
Dictionary<string, string> englishHashDictionary = new Dictionary<string,
string>();
```

El código debe presentar el resultado en el `toolStripStatusLabel2`

```
private void dictionaryToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Dictionary<string, string> englishHashDictionary = new
Dictionary<string, string>();
            DateTime startDate = DateTime.Now;
            foreach (string line in
System.IO.File.ReadLines(dictionaryFile))
            {
                if (!englishHashDictionary.ContainsKey(line))
                {
                    englishHashDictionary.Add(line, line);
                }
            }
            TimeSpan timeDiff = DateTime.Now - startDate;
            string result = string.Format("{0} add to Hashtable in
{1} ms", dictionaryFile, timeDiff.TotalMilliseconds);
            toolStripStatusLabel2.Text = result;
            //Read target file to spell-check and put not fouded
words in a list
            IList<string> missedWords = new List<string>();
            foreach (string line in
System.IO.File.ReadLines(targetFile))
            {
                string[] words = line.Replace("'", "").Replace(",",
"").Replace("!", "").Split(' ');
                foreach (string word in words)
```

```
                {
                    if (!englishHashDictionary.ContainsKey(word))
                    {
                        if (!missedWords.Contains(word))
                        {
                            missedWords.Add(word);
                        }
                    }
                }
            }
            if (missedWords.Count > 0)
            {
                MessageBox.Show(string.Join(Environment.NewLine,
missedWords));
                System.IO.File.WriteAllLines(missedWordsFile,
missedWords);
            }
        }
```

Pantalla evidencia de la ejecución del programa:



5. Codificación de la opción SortedDictionary

Basándonos en la opción anterior colocar el código necesario para usar un diccionario:

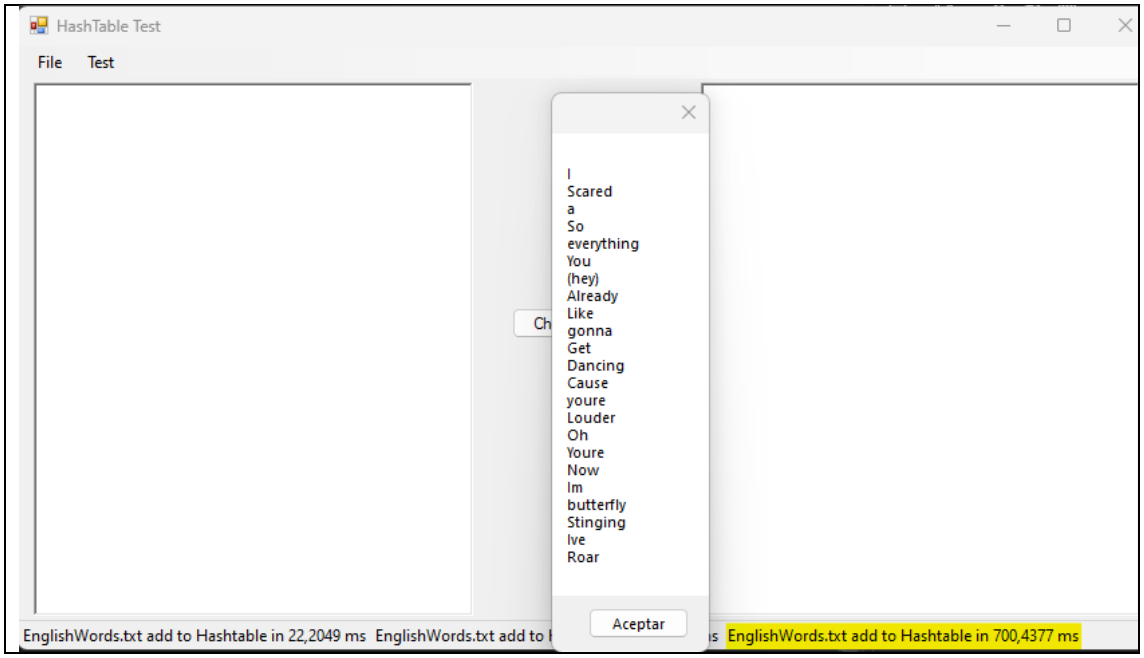
La declaración del diccionario puede ser:

```
SortedDictionary<string, string> englishHashDictionary = new
SortedDictionary<string, string>()
```

El código debe presentar el resultado en el `toolStripStatusLabel3`

```
private void sortedDirctonaryToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            //Crear un SortedDictionary para almacenar las palabras.
            SortedDictionary<string, string> englishHashDictionary = new
SortedDictionary<string, string>();
            //Leer el archivo palabra por palabra y almacenarlo en el
SortedDictionary.
            DateTime startDate = DateTime.Now;
            foreach (string line in
System.IO.File.ReadLines(dictionaryFile))
            {
                if (!englishHashDictionary.ContainsKey(line))
                {
                    englishHashDictionary.Add(line, line);
                }
            }
            TimeSpan timeDiff = DateTime.Now - startDate;
            string result = string.Format("{0} add to Hashtable in {1}
ms", dictionaryFile, timeDiff.TotalMilliseconds);
            toolStripStatusLabel3.Text = result;
            //Read target file to spell-check and put not fouded words in
a list
            IList<string> missedWords = new List<string>();
            foreach (string line in System.IO.File.ReadLines(targetFile))
            {
                string[] words = line.Replace("'", "").Replace(",",
"").Replace("!", "").Split(' ');
                foreach (string word in words)
                {
                    if (!englishHashDictionary.ContainsKey(word))
                    {
                        if (!missedWords.Contains(word))
                        {
                            missedWords.Add(word);
                        }
                    }
                }
            }
            if (missedWords.Count > 0)
            {
                MessageBox.Show(string.Join(Environment.NewLine,
missedWords));
                System.IO.File.WriteAllLines(missedWordsFile,
missedWords);
            }
        }
```

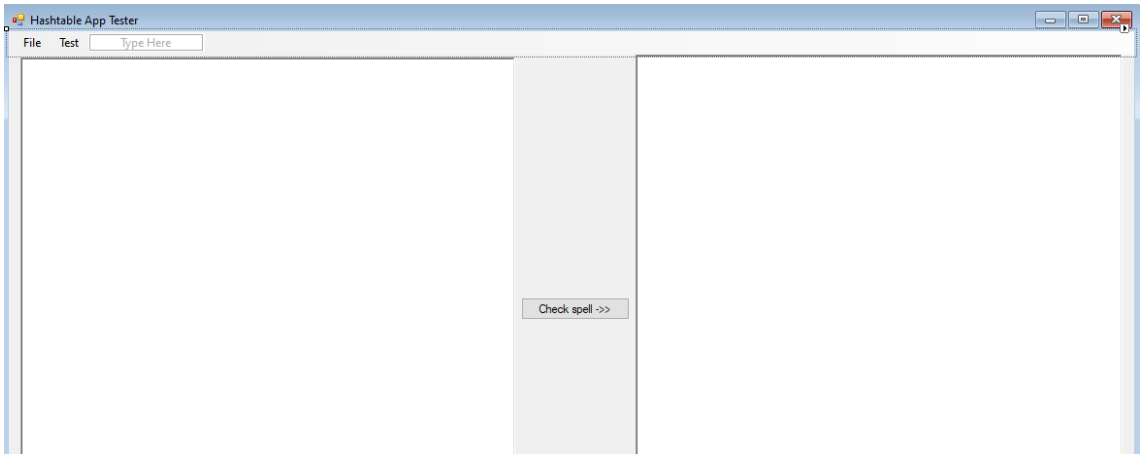
Pantalla evidencia de la ejecución del programa:



6. Diseño y codificación de pantalla principal

Diseñar una pantalla principal usando los siguientes elementos:

- **richTextBox**
- **button**



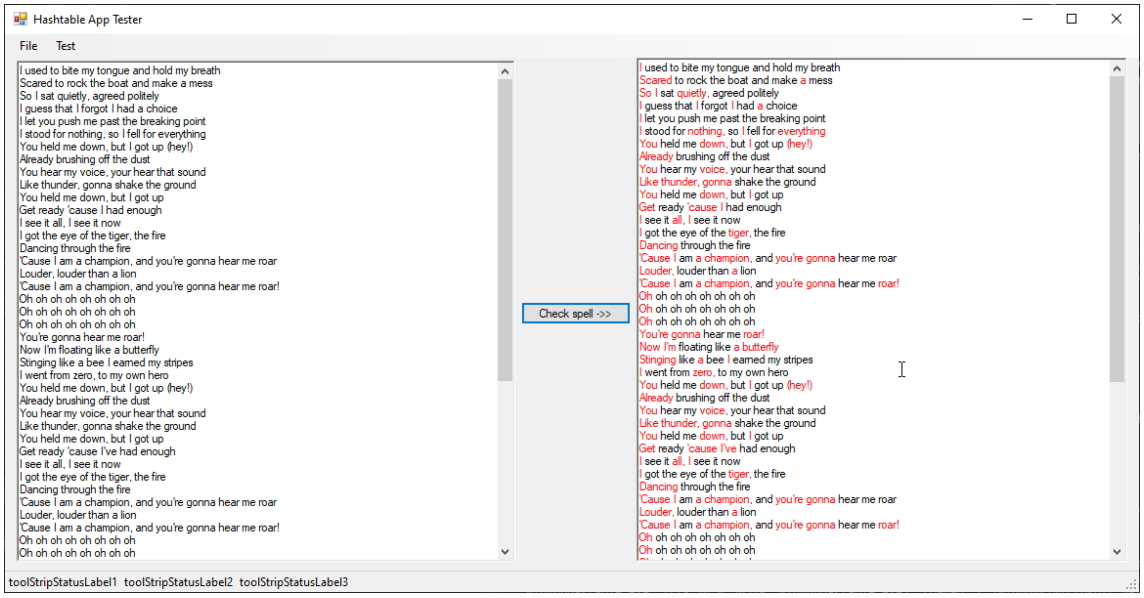
Al usar la opción Open File se puede tener el siguiente código sugerido:

```
        private void openFileToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog())
            {
                //openFileDialog.InitialDirectory = "c:\\";
                openFileDialog.Filter = "txt files (*.txt)|*.txt|All files
(*.*)|*.*";
                openFileDialog.FilterIndex = 2;
                openFileDialog.RestoreDirectory = true;

                if (openFileDialog.ShowDialog() == DialogResult.OK)
```

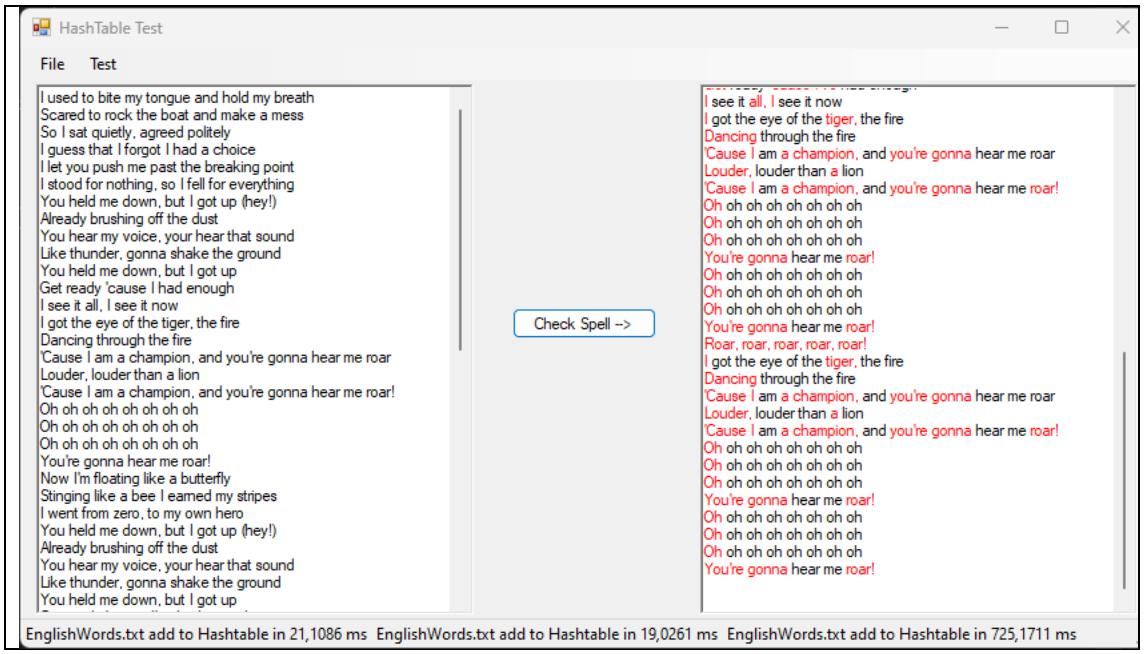
```
                {
                    //Get the path of specified file
                    filePath = openFileDialog.FileName;
                    richTextBoxFileInitial.LoadFile(filePath,
RichTextBoxStreamType.PlainText);
                }
            }
        }
```

Finalmente programar el código necesario para que al presionar el botón, pueda colocar las letras con rojo no encontradas. El resultado será similar al siguiente:




```
private void button1_Click(object sender, EventArgs e)
        {
            Hashtable englishHashDictionary = new Hashtable();
            foreach (string line in
System.IO.File.ReadLines(dictionaryFile))
            {
                if (!englishHashDictionary.ContainsKey(line))
                {
                    englishHashDictionary.Add(line, line);
                }
            }
            // Compara cada palabra de la canción con el diccionario
            foreach (string line in tA1.Lines)
            {
                string[] words = line.Split(' ');
                foreach (string word in words)
                {
                    if (!englishHashDictionary.ContainsKey(word))
                        tA2.SelectionColor = Color.Red;
                    else
                        tA2.SelectionColor = Color.Black;
                    tA2.AppendText(word + " ");
                }
                tA2.AppendText(Environment.NewLine);
            }
        }
```

Pantalla evidencia de la ejecución del programa:



7. Subir el documento actualizado en la plataforma.

Finalmente, subimos este documento actualizado como evidencia a la Plataforma del curso.