Grupo Weboo
Universidad de La Habana

Examen Final de Programación
Curso 2017-2018

Telégrafo Roto

NOTA: Si usted está leyendo este documento sin haber extraído el compactado que se le entregó, ciérrelo ahora, extraiga todos los archivos en el escritorio, y siga trabajando desde ahí. Es un error común trabajar en la solución dentro del compactado, lo cual provoca que los cambios no se guarden. Si usted comete este error y entrega una solución vacía, no tendrá oportunidad de reclamar.

El **Telégrafo** es un dispositivo que emplea señales eléctricas para la transmisión de mensajes de texto codificados, como con el código Morse, mediante líneas alámbricas o comunicaciones de radio. La Facultad de MATCOM cuenta con un viejo telégrafo con desperfectos. Este no se utiliza debido a que los mensajes recuperados no contienen la separación entre las letras.

Por ejemplo, en el código Morse, cada letra del alfabeto se sustituye por una secuencia de puntos y rayas de la siguiente manera (*Figura 1*):

Figura 1: Código Morse

Si los espacios entre las letras se pierden, los mensajes pueden ser ambiguos. Por ejemplo, incluso si sabemos que el mensaje **-..-----.** se compone de tres letras, aún podría significar: **njg**, **dog**, **xmg** o **xon**.

Usted deberá crear un programa que lea en un mensaje codificado y se obtengan todas las posibles interpretaciones de este mensaje.

Implemente el método `IEnumerable<string> DecodificarMensaje(Dictionary<char, string> alfabeto, string mensaje)` que produce todas las posbibles interpretaciones del mensaje, dado un alfabeto de codificación.

Usted debe haber recibido junto a este documento una solución de Visual Studio con dos proyectos: una biblioteca de clases (*Class Library*) y una aplicación de consola (*Console Application*). Deberá implementar el método `DecodificarMensaje` que se encuentra en la clase `TelegrafoRoto` en el *namespace* `Weboo.Examen`. En la biblioteca de clases encontrará la siguiente definición:

```
namespace Weboo.Examen
{
    public class TelegrafoRoto
    {
        public static IEnumerable<string> DecodificarMensaje(Dictionary<char, string>
alfabeto, string mensaje)
        {
            //Borre la siguiente línea y escriba su código
            throw new NotImplementedException();
        }
    }
}
```

Este método recibe como parámetros:

- `Dictionary<char, string> alfabeto`: Un diccionario que representa el código de codificación. Cada llave es un valor tipo char que es una letra y cada valor asociado a la llave es un string que corresponde al código de dicha letra. Se garantiza (no tiene que chequearlo) que dicho string solo contienes caracteres "**.**" (punto) y "**-**" (guión).
- `string mensaje`: Una cadena que contiene una secuencia de símbolos . (punto) y – (guión) que representan una palabra codificada.

El método que usted debe implementar tiene que computar un `IEnumerable<string>` que contenga **todas las posibles palabras** que se pueden obtener al decodificar el mensaje. **Si no es posible decodificar el mensaje** entonces el método debe devolver `null`.

NOTA: Todo el código de la solución debe estar en este proyecto (biblioteca de clases), pues es el único código que será evaluado. Usted puede adicionar todo el código que considere necesario, pero no puede cambiar los nombres del namespace, clase o método mostrados. De lo contrario, el probador automático fallará. En particular, es imprescindible que usted no cambie los parámetros del método `DecodificarMensaje`, ni su orden. Por supuesto, usted puede (y debe) adicionar todo el código que necesite.

Ejemplos

Utilizando como alfabeto, la codificación en código morse:

```
Dictionary<char, string> alfabetoMorse = new Dictionary<char, string>
{
    {'a',".-"},  {'b',"-..."}, {'c',"-.-."},  {'d',"-.."},   {'e',"."},
    {'f',"..-."},  {'g',"--."},      {'h',"...."},  {'i',".."},    {'j',".---"},
    {'k',"-.-"},   {'l',".-.."}, {'m',"--"},    {'n',"-."},    {'o',"---"},
    {'p',".--."},  {'q',"--.-"}, {'r',".-."},   {'s',"..."},   {'t',"-"},
    {'u',"..-"},   {'v',"...-"}, {'w',".--"},   {'x',"-..-"}, {'y',"-.--"},
    {'z',"--.."}
};

//Ejemplo #1
string mensaje1 = "..-";
IEnumerable<string> resultado1 = TelegrafoRoto.DecodificarMensaje(alfabetoMorse,
mensaje1);
// resultado1 => "u", "ea", "it", "eet"
```

```
//Ejemplo #2
string mensaje2 = "...";
IEnumerable<string> resultado2 = TelegrafoRoto.DecodificarMensaje(alfabetoMorse,
mensaje2);
// resultado2 => "eee", "ei", "ie", "s"
```

Sin embargo, con este otro código de codificación se obtienen diferentes resultados:

```
Dictionary<char, string> alfabetoHawaiano = new Dictionary<char, string>
{
    { 'a', "..-" }, { 'e', ".-.-." }, { 'h', "--.-" }, { 'i', ".-" },
    { 'k', "--." }, { 'l', "--" }, { 'm', "-.-.-" }, { 'n', "-.-" },
    { 'o', "." }, { 'p', "-.." }, { 'u', ".-." }, { 'w', ".-.-." }
};

//Ejemplo #3
string mensaje3 = "-";
IEnumerable<string> resultado3 = TelegrafoRoto.DecodificarMensaje(alfabetoHawaiano,
mensaje3);
// resultado3 => null
```

NOTA: Los casos de prueba que aparecen en este proyecto son solamente de ejemplo. Que usted obtenga resultados correctos con estos casos no es garantía de que su solución sea correcta y de buenos resultados con otros ejemplos. De modo que usted debe probar con todos los casos que considere convenientes para comprobar la validez de su implementación.