MATCOM

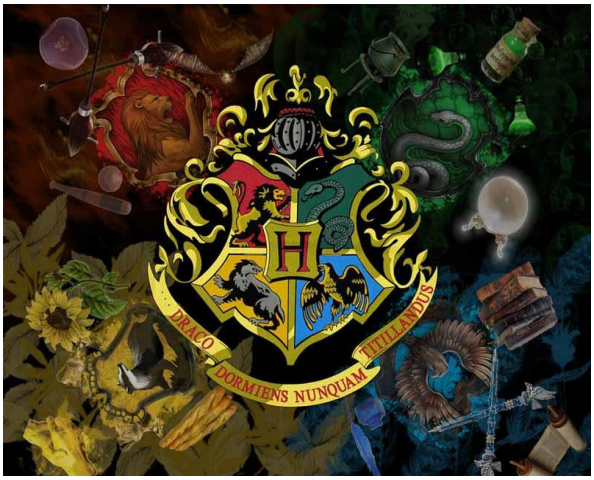
Grupo Weboo
Universidad de La Habana

Segundo Examen de Programación

Curso 2019-2020

Tom Riddle

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.



Han pasado años desde la última derrota de los mortífagos en Hogwarts y la supuesta muerte de su líder Lord Voldemort a manos de Harry Potter y la orden del Fénix. Luego de una época de tranquilidad y prosperidad en el mundo de los magos, se han iniciado una serie de sucesos preocupantes que apuntan a una posible resurrección de "*el que no debe ser nombrado*". Harry y sus amigos han acudido al Ministro de Magia para alertar la situación pero una vez más han sido ignorados. Es por eso que necesitan tu ayuda.

Harry sospecha que esta vez Voldemort y sus mortífagos están usando nuevos nombres para esconder su identidad, pero que estos nuevos nombres son anagramas de los anteriores, tal como sucedió anteriormente con "Tom Marvolo Riddle" y "I am Lord Voldemort" (que se traducide a "yo soy Lord Voldemort").

Se dice que una palabra es anagrama de otra si las dos tienen las mismas letras, con el mismo número de apariciones.

Ejemplo:

- lectura -> cuartel
- amor -> Roma
- reconquistados -> conquistadores

Harry conoce el nombre real de cada mortífago y para cada uno de ellos ha logrado reducir la búsqueda de su nueva identidad a una lista de posibles palabras que combinadas podrían formar un anagrama del nombre original.

Usted debe, dado un string (nombre real) y un array de strings (posibles secciones del anagrama), hallar de cuántas formas se pueden construir anagramas del nombre real a partir de la concatenación de uno o más strings del array.

Todos los strings estarán compuestos por caracteres de la ‘a’ a la ‘z’ (siempre en minúscula).

Ejemplo 1

Supongamos que el nombre original es “riddle” y las posibles secciones del anagrama son {“lid”,”tomas”,”red”}.

Entonces se podrían construir **dos** anagramas: “redlid” y “lidred” (a partir de ‘lid’ y ‘red’)

Ejemplo 2

Supongamos que el nombre original es “denver” y las posibles secciones del anagrama son {“enver”,”ner”,”ved”,”ren”, “edv”}.

Entonces se podrían construir los anagramas:

- “nerved” y “vedner” (a partir de “ner” y “ved”)

- “neredv” y “edvner” (a partir de “ner” y “edv”)

- “vedren” y “renved” (a partir de “ved” y “ren”)

- “renedv” y “edvren” (a partir de “ren” y “edv”)

La respuesta sería 8.

Ejemplo 3

Supongamos que el nombre original es “rama” y las posibles secciones del anagrama son {“a”,”rm”,”a}.

Entonces se podrían construir los anagramas:

- “arma”, “arma”

- “rmaa”, “rmaa”

- “aarm”, “aarm”

La respuesta sería 6.

Tener en cuenta que cómo lo que se desea contar es la cantidad de formas de construir los anagramas entonces, si es posible construir el mismo anagrama de más de una forma, se debe contar cada una de las formas de construirlo, tal como sucede en el ejemplo anterior.

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*). Usted debe implementar el método `CantidadDeAnagramas` que se encuentra en la clase `Anagrama` en el *namespace* `Weboo.Examen`. En la biblioteca de clases encontrará la siguiente definición:

```
namespace Weboo.Examen
{
    public class Anagrama
    {
        public static int CantidadDeAnagramas(string nombre, string[] secciones)

        {
            //Borre la siguiente línea y escriba su código
            throw new NotImplementedException();
        }
    }
}
```

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 el parámetro del método CantidadDeAnagramas. Por supuesto, usted puede (y debe) adicionar todo el código que necesite.

Ejemplos

```
//Ejemplo 1

    string nombre1 = "Riddle";
    string[] substr1 = { "lid", "tomas", "red" };
    int resp = Anagrama.CantidadDeAnagramas(nombre1, substr1);
    Test(1, 2, resp);

    //SOLUCION -> 2


    //Ejemplo 2

    string nombre2 = "denver";
    string[] substr2 = { "enver", "ner", "ved", "ren", "edv" };
    int resp2 = Anagrama.CantidadDeAnagramas(nombre2, substr2);
    Test(2, 8, resp2);

    //SOLUCION -> 8
```

```
//Ejemplo 3

string nombre3 = "casemiro";
string[] substr3 = { "miro", "cas", "mie", "aro", "esaca" };
int resp3 = Anagrama.CantidadDeAnagramas(nombre3, substr3);
Test(3, 0, resp3);

//SOLUCION -> 0



//Ejemplo 4

string nombre4 = "rama";
string[] substr4 = { "a", "rm", "a" };
int resp4 = Anagrama.CantidadDeAnagramas(nombre4, substr4);
Test(4, 6, resp4);

//SOLUCION -> 6
```

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.