Universidad del Istmo de Guatemala
Facultad de Ingenieria
Ing. en Sistemas
Informatica 2
Prof. Ernesto Rodriguez - erodriguez@unis.edu.gt

Hoja de trabajo #5

Fecha de entrega: 1 de Marzo, 2018 - 11:59pm

Instrucciones: Realizar cada uno de los ejercicios siguiendo sus respectivas instrucciones. El trabajo debe ser entregado a traves de Github, en su repositorio del curso, colocado en una carpeta llamada "Hoja de trabajo 5". Al menos que la pregunta indique diferente, todas las respuestas a preguntas escritas deben presentarse en un documento formato pdf, el cual haya sido generado mediante Latex. Los ejercicios de programación deben ser colocados en una carpeta llamada "Programas", la cual debe colocarse dentro de la carpeta correspondiente a esta hoja de trabajo.

Reduce

La función de orden superior más utilizada probablemente es la función *Reduce* (tambien conocida como *fold*). En esta tarea se implementara dicha función y se llevara a cabo paso a paso.

Para empezar crear una solución dentro de la carpeta "Programas" y dentro de ella crear dos proyectos: "Reduce" y "ReduceTests". El proyecto "Reduce" debe ser de tipo *console* y "ReduceTests" de tipo *xunit*.

Parte 1 (50%)

En la clase "Program" del archivo "Program.cs", definir un metodo estatico llamado "ReduceInt". Este metodo tiene tipo "`ReduceInt` : `int`[] $\otimes$ `int` $\otimes$ (`int` $\otimes$ `int` $\rightarrow$ `int`) $\rightarrow$ `int`", en otras palabras:

- Acepta un arreglo de `int` como primer parametro
- Acepta un `int` como segundo parametro
- Acepta una función que toma dos `int` y produce un `int` como tercer parametro.
- Retorna un `int`

La función opera de la siguiente manera:

1. Inicializa una variable de tipo `int` con el valor del segundo parametro. Esta variable se llama *acumulador*
2. Luego un ciclo recorre todos los elementos.
3. En cada iteración del ciclo, el *acumulador* y el elemento actual se utilizan para llamar la función en el tercer parametro (conocida como la *reducción*), y el resultado de dicha función se vuelbe el nuevo valor del *acumulador*

4. Luego de recorrer todos los elementos, el resultado de reduce es el ultimo valor del *acumulador*

Ejemplos:

```
public class Program{

    public static void Main(string[] args){
        int[] valores = new int[]{1,2,3,4,5};

        // Contar elementos en un arreglo mediante
        // reduce
        Func<int,int,int> reduccion = (acc, valor) => acc + 1;
        Program.ReduceInt(valores, 0, reduccion); // Produce 5

        // Sumar los elementos de un arreglo mediante
        // reduce
        reduccion = (acc, valor) => acc + valor;
        Program.ReduceInt(valores, 0, reduccion); // Produce 15

        // Buscar el elemento mas grande en un
        // arreglo mediante reduce
        reduccion = (acc, valor) => {
            if(valor > acc){
                return valor;
            }else{
                return acc;
            }
        };
        Program.ReduceInt(valores, int.MinValue, reduccion); // Produce 5
    }
}
```

Como primer ejercicio, implementar la funcion “ReduceInt” e implementar 1 prueba unitaria (en el proyecto “ReduceTests”) que verifique el comportameinto correcto de “ReduceInt”.

Parte 2 (50%)

La forma más general de la función *Reduce*, es una version con dos parametros genericos, los llamaremos `T` y `A`. El primer parametro corresponde al tipo del arreglo que recibe la función *Reduce*, el segundo corresponde al tipo del *accumulador*. Esto conlleva a que la función tenga el tipo "$\texttt{Reduce}\langle \texttt{T}, \texttt{A} \rangle \ : \ \texttt{T}[] \otimes \texttt{A} \otimes (\texttt{A} \otimes \texttt{T} \rightarrow \texttt{A}) \rightarrow \texttt{A}$". Esta version puede trabajar con arreglos y acumuladores de tipo diferentes, dando la oportunidad de implementar una variedad de funciones. Por ejemplo:

```
public class Program{

    public static void Main(string[] args){
        string[] valores = new string[]{"Reduce", "es", "fold"};

        // Concatenar un array de strings mediante
        // commas
        Func<string,string,string> reduccion1 = (acc, valor) => {
            if(acc == string.Empty){
                return valor;
            }else{
                return acc + ", " + valor;
            }
        };
        Program.Reduce<string, string>(valores, string.Empty, reduccion1);
        // Produce "Reduce, es, fold"


        // Contar las letras en un array de strings
        Func<int, string, int> reduccion2 = (acc, valor) => acc + valor.Length;
        Program.Reduce<string, int>(valores, 0, reduccion2); // Produce 10
    }
}
```

Como segundo ejercicio debe implementar la función "Reduce" e implementar una prueba unitaria (en el proyecto "ReduceTests") que verifique el comportamiento de "Reduce".