

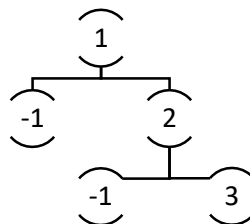
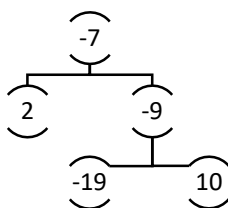
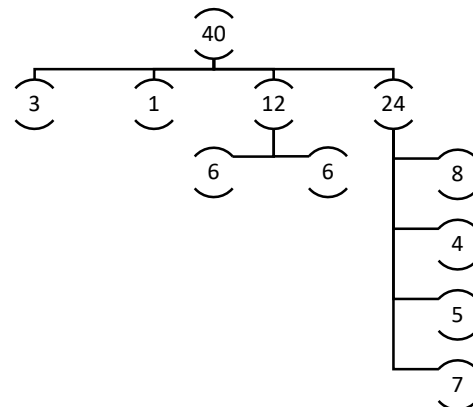
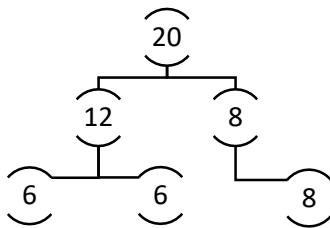
Tercer Examen de Programación  
Curso 2017-2018

## Árboles de Suma

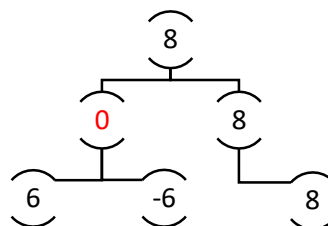
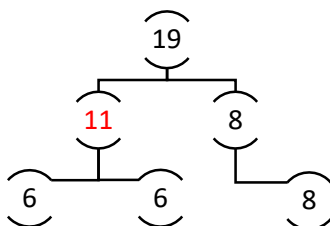
**NOTA:** Si usted está leyendo este documento sin haber extraído el compactado que se le entregó, ciérrelo ahora, extraiga todos los archivos en el escritorio, y siga trabajando desde ahí. Es un error común trabajar en la solución dentro del compactado, lo cual provoca que los cambios no se guarden.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

Un árbol de suma es una estructura arborea en la cual todos los nodos contienen valores enteros y el valor de cada nodo es la suma de los valores de sus hijos, exceptuando las hojas que pueden tener valores arbitrarios. Este tipo de estructura puede tener cantidades diferentes de hijos por nodo. Un nodo de valor cero en un árbol de suma no puede existir, exceptuando que sea el árbol de suma con un único nodo raíz con valor 0.

Algunos ejemplos válidos de árboles de suma son presentados a continuación:



A continuación se muestran ejemplos de árboles de suma **INVÁLIDOS**:



Usted debe implementar varias operaciones definidas sobre los árboles de suma a través de la implementación de las interfaces `IArbolSuma` e `ITree<T>`.

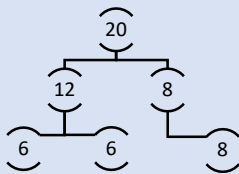
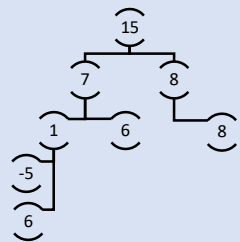
```
public interface IArbol<T>
{
    T Valor { get; } // Valor del nodo
    IEnumerable<IArbol<T>> Hijos { get; } // Hijos
    IEnumerable<T> PostOrden(); // Recorrido PostOrden
    IEnumerable<T> PreOrden(); // Recorrido PreOrden
}

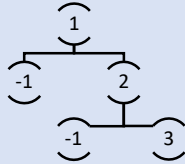
public interface IArbolSuma : IArbol<int>
{
    void InsertarPreOrden(int k, IArbolSuma nuevo);
    void InsertarPostOrden(int k, IArbolSuma nuevo);
    void InsertarKHoja(int k, IArbolSuma nuevo)
}
```

Los métodos `InsertarPreOrden` e `InsertarPostOrden` insertan un árbol suma pasado como parámetro como hijo del k-ésimo nodo del pre-orden o post-orden respectivamente manteniendo la invariante del árbol de suma: **todos los nodos son la suma de sus hijos y no debe existir ningún nodo de valor 0, exceptuando el caso raíz**. De igual manera `InsertarKHoja`, insertará el nodo nuevo como hijo de la k-ésima hoja del árbol de izquierda a derecha realizando las actualizaciones necesarias para mantener el estado del árbol correctamente.

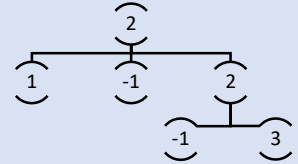
En el caso de que la inserción se realice en una hoja, esta se transformará en nodo, su valor anterior se guardará en una nueva hoja hija de ella y se añadirá el nodo a insertar posteriormente. Se deberá posteriormente actualizar el estado del árbol.

En caso de que no exista el k-ésimo elemento para cualquiera de estos métodos, el árbol permanecerá sin modificaciones. Ejemplos de estas operaciones son mostradas a continuación:

ArbolSuma a	Operación	Resultado
	<pre>a.InsertarPreOrden (2, new ArbolSuma (-5))</pre>	
<p><b>Descripción del Ejemplo:</b></p> <p>El nuevo valor se insertará como hijo del tercer nodo en el preorden es decir <b>como hijo de la hoja más a la izquierda con valor (6)</b>. La hoja mencionada se convierte en nodo, se crea una hoja con su valor previo (6) y se añade el nodo a insertar (-5). Se actualiza el árbol para mantener un estado correcto.</p>		

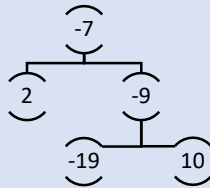


a.InsertarPreOrden  
(0, new ArbolSuma (1))

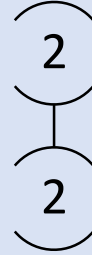


#### Descripción del Ejemplo:

El nuevo valor se insertará como hijo del primer nodo en el preorden es decir como hijo de la raíz, que tiene valor (1). Se actualiza el árbol para mantener un estado correcto.

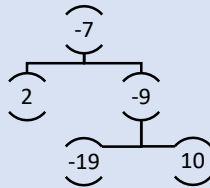


a.InsertarPreOrden  
(2, new ArbolSuma (9))

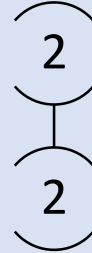


#### Descripción del Ejemplo:

El nuevo valor se insertará como hijo del tercer nodo en el preorden es decir como hijo del nodo con valor (-9). Se actualiza el árbol para mantener un estado correcto. El nodo (-9) pasa a sumar 0, por lo cual es eliminado todo su subarbol.

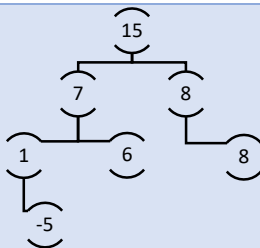


a.InsertarPostOrden  
(3, new ArbolSuma (9))



#### Descripción del Ejemplo:

Idem al anterior, pero con post-orden.

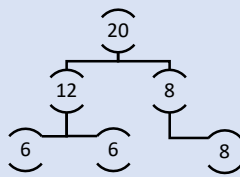


a.InsertarPostOrden  
(6, new ArbolSuma (-15))

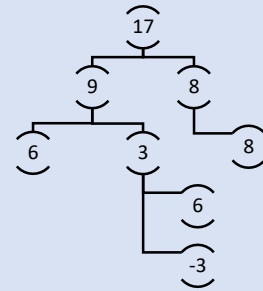


#### Descripción del Ejemplo:

El nuevo valor se insertará como hijo del séptimo nodo en el post-orden es decir como hijo de la raíz, que tiene valor (15). Se actualiza el árbol para mantener un estado correcto. La raíz se queda con valor (0), se eliminan todos los nodos exceptuando el raíz.

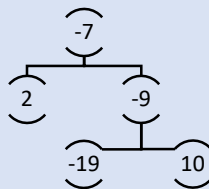


a.InsertarKHoja  
(1, new ArbolSuma (-3))

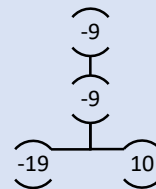


#### Descripción del Ejemplo:

El nuevo valor se insertará como hijo de la segunda hoja con valor (6). La hoja mencionada se convierte en nodo, se crea una hoja con su valor anterior y se añade el nuevo nodo (-3). Se actualiza el árbol para mantener un estado correcto.

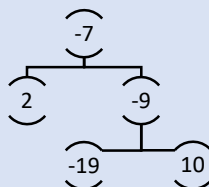


a.InsertarKHoja  
(0, new ArbolSuma (-2))



#### Descripción del Ejemplo:

El nuevo valor se insertará como hijo de la primera hoja con valor (2). La hoja mencionada se convierte en nodo, se crea una hoja con el valor anterior del nuevo nodo (2), se añade la nueva hoja (-2). Al actualizar los valores, la suma da 0, por lo cual el hijo izquierdo de la raíz es eliminado. Se actualiza el árbol para mantener un estado correcto.



a.InsertarKHoja  
(0, new ArbolSuma (7))



#### Descripción del Ejemplo:

Se crea un nodo con valor (7). Este se inserta como hijo de la primera hoja con valor (2). La hoja mencionada se convierte en nodo y toma valor (9), suma de la hoja creada a partir de su antiguo valor (2) y la insertada con valor (7). Se actualiza el árbol para mantener un estado correcto. La raíz se queda con valor 0, se eliminan todos los nodos exceptuando el raíz.

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 completar la implementación de la clase `ArbolSuma` en el *namespace* `Weboo`. Examen que ya implementa la interface `IArbolSuma`. En la biblioteca de clases encontrará la siguiente definición:

```
class ArbolSuma : IArbolSuma
{
    // !!! NO MODIFIQUE/ELIMINE LA DECLARACION DEL CONSTRUCTOR, PUES HARA QUE SU CODIGO NO
    // !!! FUNCIONE CON EL TESTER Y TENDRA AUTOMATICAMENTE SUSPENSO EL EXAMEN
    public ArbolSuma(params int[] values)
    {
        throw new System.NotImplementedException();
    }

    public int Valor
    {
        get
        {
            throw new System.NotImplementedException();
        }
    }

    public IEnumerable<IArbol<int>> Hijos
    {
        get
        {
            throw new System.NotImplementedException();
        }
    }

    public IEnumerable<int> PostOrden()
    {
        throw new System.NotImplementedException();
    }

    public IEnumerable<int> PreOrden()
    {
        throw new System.NotImplementedException();
    }

    public void InsertarPreOrden(int k, IArbolSuma nuevo)
    {
        throw new System.NotImplementedException();
    }

    public void InsertarPostOrden(int k, IArbolSuma nuevo)
    {
        throw new System.NotImplementedException();
    }

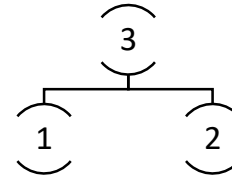
    public void InsertarKHoja(int k, IArbolSuma nuevo)
    {
        throw new NotImplementedException();
    }
}
```

Como se puede observar el constructor recibe una cantidad variable de argumentos. En caso de que se pase un solo valor, se debe crear una hoja con ese valor. En caso contrario se debe crear un nodo con la misma cantidad de hojas que valores pasados al constructor, tomando cada hoja el valor correspondiente de los parámetros. A continuación se muestran algunos ejemplos de creación de árboles de suma:

`new ArbolSuma(2)`



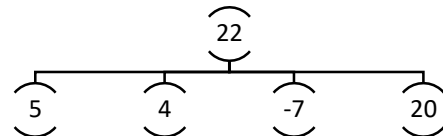
`new ArbolSuma(1,2)`



`new ArbolSuma(-2,2)`



`new ArbolSuma(5,4,-7,20)`



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, ni la signature del constructor de `ArbolSuma`. De lo contrario, el probador automático fallará.

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.