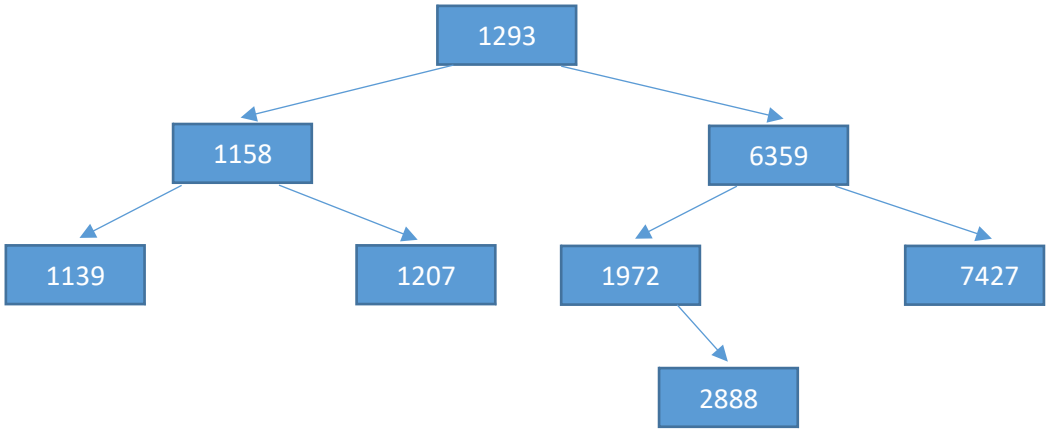
Hoja de trabajo

Parte 1: Eliminación en árboles binarios de búsqueda

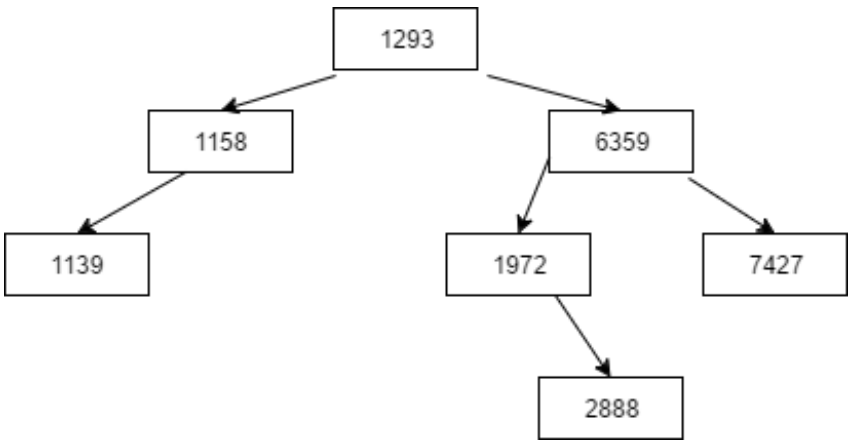
Material de referencia:

- Lectura árboles binarios de búsqueda.
  Libro: "*Data Structures and Program Design in C++*" de Kruse y Ryba.
  Páginas: 444-462.

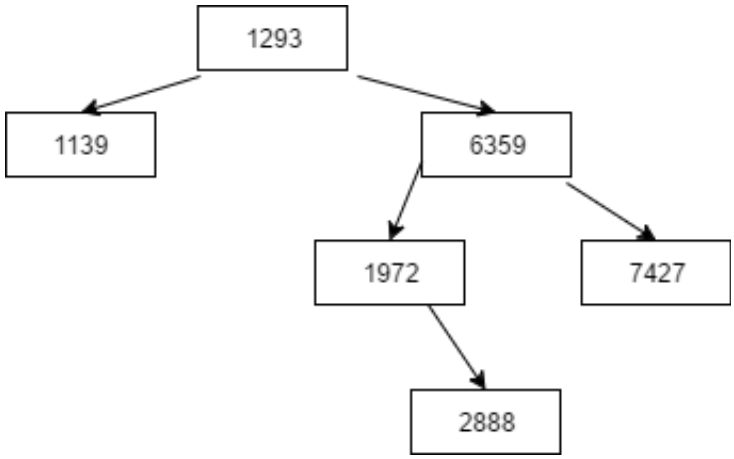
Se tiene el siguiente árbol binario de búsqueda, el cual guarda números de participantes de un torneo de natación. Cada atleta se identifica por su número de participante.



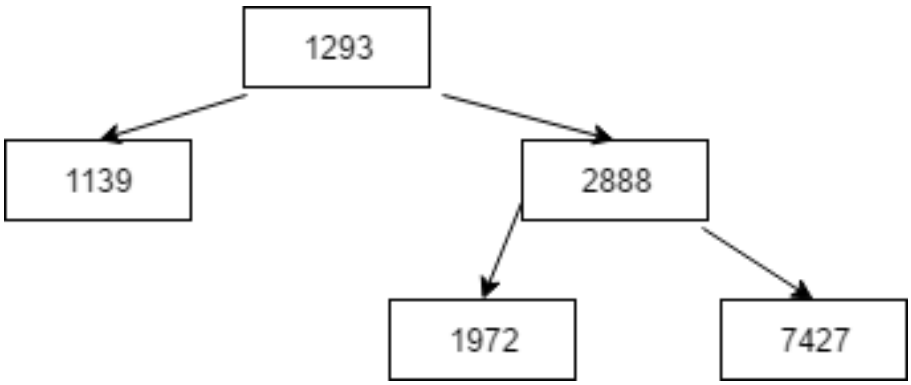
1. El participante No. 1207 por una lesión decide retirarse del torneo. Dibuje el árbol resultante después de la eliminación del nodo cuyo dato es 1207.



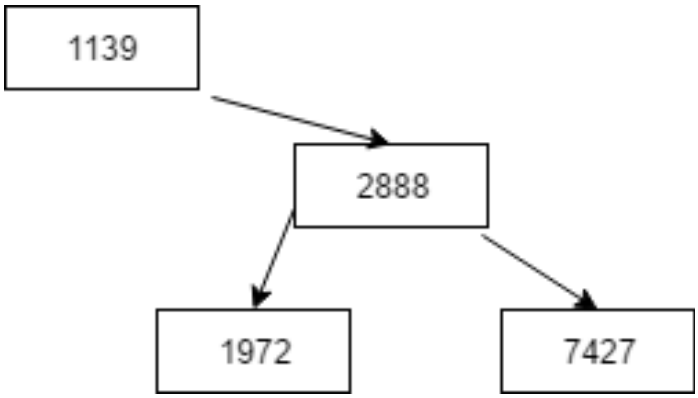
2. El participante No. 1158 queda descalificado por incumplir una norma. Se debe eliminar este nodo, pero ¿qué se debe hacer con el nodo No. 1139? Recuerde que este participante todavía sigue en la competencia. Dibuje el árbol resultante.



3. El participante No. 6359 queda eliminado después de una ronda. En este árbol binario de búsqueda se desea mantener únicamente los atletas que siguen activos en el torneo, por lo que este nodo deberá ser eliminado. ¿Qué se debe hacer los nodos No. 1972, 2888 y 7427? Dibuje el árbol resultante.



4. A veces es necesario tomar decisiones difíciles, por una emergencia familiar, el participante No. 1293 decide retirarse de la competencia. Dibuje el árbol resultante.

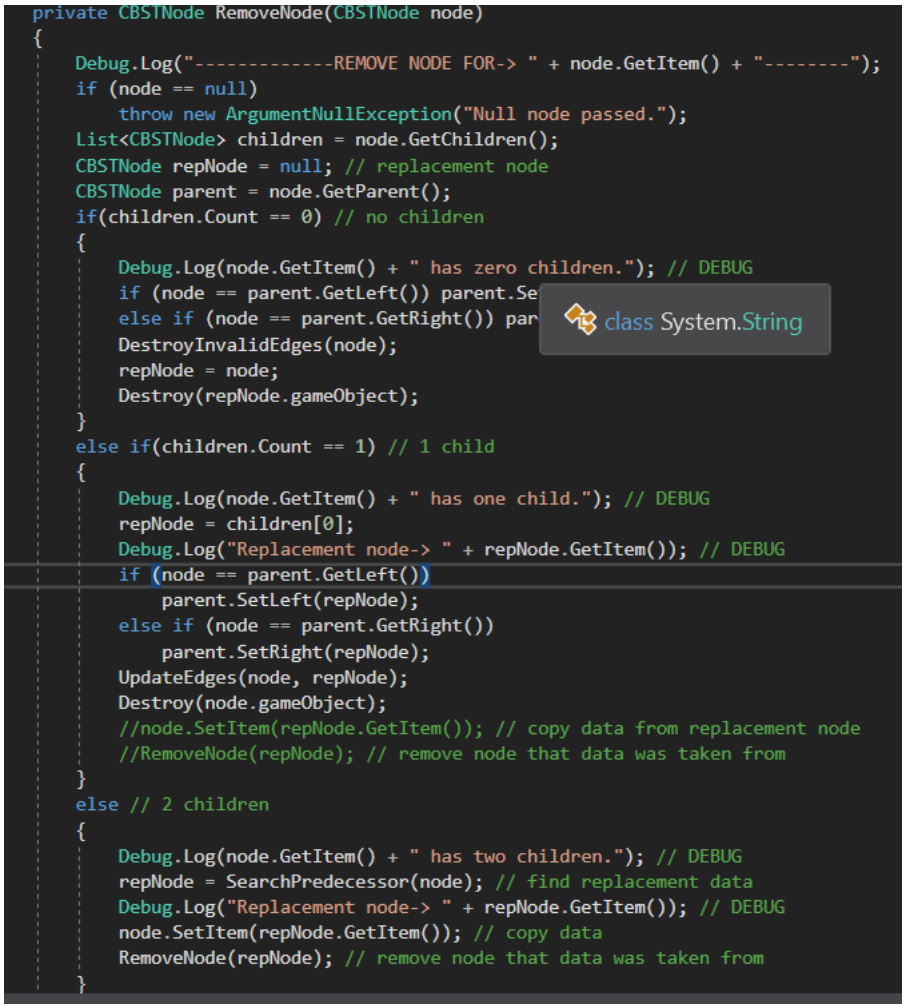


5. A usted se le pide que le explique a un alumno de Estructuras de Datos, que le explique el procedimiento para eliminar un nodo de árbol binario. ¿Qué le diría?
    - Eliminar un nodo de un árbol binario no es tan sencillo como eliminar un nodo de una lista enlazada. Tenemos que hay 3 casos: el nodo no tiene hijos, el nodo tiene 1 hijo, el nodo tiene 2 hijos. Cuando el nodo no tiene hijo, simplemente lo eliminamos ajustando las referencias del padre y del hijo. Cuando tiene un hijo, debemos hacer que ahora el padre del nodo a eliminar apunte al hijo del nodo a eliminar. Cuando tiene 2 hijos hay varias maneras. Una de ellas es buscar al hijo pequeño más grande, y utilizar este hijo para que reemplaze al padre.
6. Escriba un algoritmo en pseudocódigo para eliminar un nodo de árbol binario de búsqueda.

ALGORITMO_eliminar

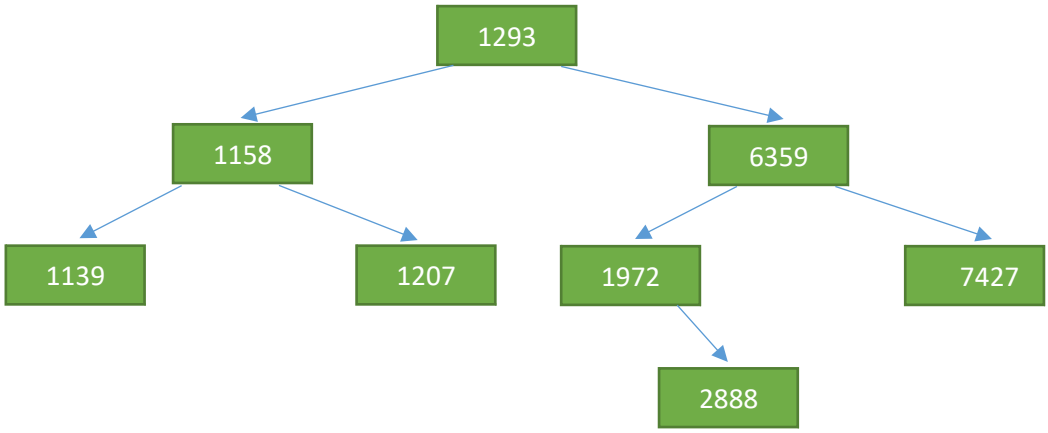
```
ALGORITMO eliminar(CNodo nodoAEliminar)

INPUT nodoAEliminar
VARIABLES
  CNodo : padre, hijo

INICIO
  SI(nodoAEliminar no tiene hijos) ENTONCES
  INICIO
    eliminar(nodoAEliminar)
  FIN
  SI(nodoAEliminar tiene 1 hijo) ENTONCES
  INICIO
    padre = nodoAEliminar.padre
    hijo = nodoAEliminar.unicoHijo
    SI(nodoAEliminar es hijo izquierdo) ENTONCES
      padre.hijoIzquierdo = hijo
    SINO
      padre.hijoDerecho = hijo
  FIN
  SINO // tiene 2 hijos
  INICIO
    reemplazo = predecesor(nodoAEliminar)
    nodoAEliminar.dato = reemplazo.dato
    eliminar(reemplazo)
  FIN
FIN
```

7. Escriba el código de C# para eliminar un nodo de árbol binario de búsqueda.



```
private CBSTNode RemoveNode(CBSTNode node)
{
    Debug.Log("------------REMOVE NODE FOR-> " + node.GetItem() + "--------");
    if (node == null)
        throw new ArgumentNullException("Null node passed.");
    List<CBSTNode> children = node.GetChildren();
    CBSTNode repNode = null; // replacement node
    CBSTNode parent = node.GetParent();
    if(children.Count == 0) // no children
    {
        Debug.Log(node.GetItem() + " has zero children."); // DEBUG
        if (node == parent.GetLeft()) parent.Se
        else if (node == parent.GetRight()) par
        DestroyInvalidEdges(node);
        repNode = node;
        Destroy(repNode.gameObject);
    }
    else if(children.Count == 1) // 1 child
    {
        Debug.Log(node.GetItem() + " has one child."); // DEBUG
        repNode = children[0];
        Debug.Log("Replacement node-> " + repNode.GetItem()); // DEBUG
        if (node == parent.GetLeft())
            parent.SetLeft(repNode);
        else if (node == parent.GetRight())
            parent.SetRight(repNode);
        UpdateEdges(node, repNode);
        Destroy(node.gameObject);
        //node.SetItem(repNode.GetItem()); // copy data from replacement node
        //RemoveNode(repNode); // remove node that data was taken from
    }
    else // 2 children
    {
        Debug.Log(node.GetItem() + " has two children."); // DEBUG
        repNode = SearchPredecessor(node); // find replacement data
        Debug.Log("Replacement node-> " + repNode.GetItem()); // DEBUG
        node.SetItem(repNode.GetItem()); // copy data
        RemoveNode(repNode); // remove node that data was taken from
    }
```

Parte 2: Recorrido árboles binarios



8. En el árbol binario de la figura anterior, si se realiza un recorrido “en orden”, indique los números de participante que serán visitados (según el orden).
   1139, 1158, 1207, 1293, 1972, 2888, 6359, 7427
9. Si se realiza un recorrido “pre orden”, indique la secuencia de números de participante que serán visitados.
   1293, 1158, 1139, 1207, 6359, 1972, 2888, 7427
10. Si se realiza un recorrido “pos orden”, indique la secuencia de números de participante que serán visitados.
    1139, 1207, 1158, 2888, 1972, 7427, 6359, 1293
11. Si se realiza un recorrido “por niveles”, indique la secuencia de números de participante que serán visitados.
    1293, 1158, 6359, 1139, 1207, 1972, 7427, 2888
12. Pasando a otro tema, en C# ¿qué es un delegado (Delegate)? Investigue y explique en un párrafo. Proporcione un ejemplo e indique cuál podría ser un uso de los delegados.
    Un delegado es un tipo de dato que representa referencias metodos. Se utilizan para pasar metodos como parametros a otros metodos.
13. ¿Recuerda el concepto de clases abstractas e interfaces? Investigue el “patrón de diseño Estrategia” y descríbalo en un párrafo. ¿Cuál podría ser un uso del patrón de estrategia?
    - Es un patrón que permite cambiar el comportamiento de un algoritmo en tiempo de ejecución.
    - Para esto: define una familia de algoritmos, los encapsula, y los hace intercambiables entre ellos.
    - No estoy totalmente seguro de que se pueda, pero podría ser posible tomar un árbol binario por ejemplo, y luego durante la ejecución del programa, cambiar a que sea un árbol binario de búsqueda.
14. Investigue: ¿Qué es una función Call Back? ¿Para qué se utiliza?
    - Un callback es cualquier bloque de código que se pasa como argumento a otro bloque de código que debe ejecutar al argumento en algún momento.
    - Me imagino que hay muchos usos, uno que encontré es para no detener la ejecución de un programa en lo que se espera que una función retorne. Por ejemplo, en una aplicación web, si esperamos a que se bajen los datos necesarios, la interfaz gráfica se quedará trabada. Si en vez definimos una

función que llame a otra función cuando los datos estén disponibles y mientras tanto se muestra un mensaje de "Descargando datos", será mucho mejor.

15. Regresando al tema de árboles, coloque los algoritmos en pseudocódigo para: recorrer el árbol en orden, pre orden, pos orden y por niveles.

```
ALGORTIMO posOrden

INICIO
  SI(nodoActual.hijoIzquierdo !=  null) ENTONCES
    visitar(nodoActual.hijoIzquierdo)
  SI(nodoActual.hijoDerecho !=  null) ENTONCES
    visitar(nodoActual.hijoDerecho)
  visitar(nodoActual)
FIN
```

```
ALGORTIMO enOrden

INICIO
  SI(nodoActual.hijoIzquierdo !=  null) ENTONCES
    visitar(nodoActual.hijoIzquierdo)
  visitar(nodoActual)
  SI(nodoActual.hijoDerecho !=  null) ENTONCES
    visitar(nodoActual.hijoDerecho)
FIN
```

```
ALGORTIMO preOrden

INICIO
  visitar(nodoActual)
  SI(nodoActual.hijoIzquierdo !=  null) ENTONCES
    visitar(nodoActual.hijoIzquierdo)
  SI(nodoActual.hijoDerecho !=  null) ENTONCES
    visitar(nodoActual.hijoDerecho)
FIN
```

```
ALGORTIMO porNiveles

VARIABLES
  COLA : cola

INICIO
  SI !(nodoActual.visitado) ENTONCES
  INICIO
    cola.Enqueue(nodoActual)
    nodoActual.visitado = true
  FIN
  SI (nodoActual.hijoIzquierdo != null) ENTONCES
  INICIO
    cola.Enqueue(nodoActual.hijoIzquierdo)
    nodoActual.hijoIzquierdo.visitado = true
    porNiveles(nodoActual.hijoIzquierdo)
  FIN
  SI (nodoActual.hijoDerecho != null) ENTONCES
  INICIO
    cola.Enqueue(nodoActual.hijoDerecho)
    nodoActual.hijoDerecho.visitado = true
    porNiveles(nodoActual.hijoDerecho)
  FIN
FIN
```

16. Código C# para recorrido por niveles.

```
private List<string> BreadthFirstTraversal(BinaryTreeNode<T> n,
    List<string> list)
{
    if(n != null)
    {
        SListQueue<BinaryTreeNode<T>> q = new SListQueue<BinaryTreeNode<T>>();
        q.Enqueue(n);
        list.Add(n.GetItem().ToString());
        n.Visit();
        while(!q.IsEmpty())
        {
            BinaryTreeNode<T> node = q.Dequeue();
            foreach (BinaryTreeNode<T> adN in Children(node))
            {
                adN.Visit();
                list.Add(adN.GetItem().ToString());
                q.Enqueue(adN);
            }
        }
    }
    return list;
}
```