Binary Search Trees (BST)

Objective

The objective of this lab is to understand Binary Search Tree using array based implementation.

Task 1

Delete a node in BST in linked list and add this method in the binary search tree implementation as you did in the last lab.

Procedure

Deleting a node is the most common operation required for binary search trees. You start by finding the node you want to delete, using the same approach as you did in find() and insert() method. once node is found then there are three possible cases in binary search tree to consider:

1. **The node to be deleted is a leaf (has no children):**
   To delete a leaf node, you simply change the appropriate child field in the node's parent to point to null, instead of the node.

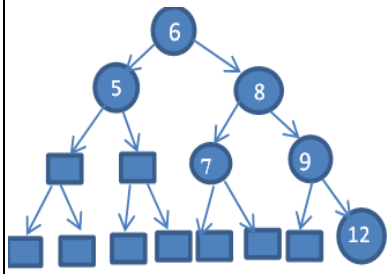
2. **The node to be deleted has one child:**
   The node has only two connections: to its parent and to it's only a child. You want to cut the node out of this sequence by connecting its parent directly to its child.

3. **The node to be deleted has two children:**
   To delete a node with two children, replace the node with minimum value of its right subtree or maximum value of its left subtree and delete the node as discuss above in case 1 or case 2.

```
Delete(Object element){
        node[] ref=find(data)  //find node to delete that return node t and its parent p references
        If(node t has no child)  {  // call deleteNochild(t,p)  }
        If(node t has one child) {  // call deleteOnechild(t,p)   }
        else{   //(node t has two children)
            Find node minNode with minimum value on t's  right subtree
           Replace t data with this minNode value
          If(minNode is noChiled node)  // remove minNode
              // call deleteNochild(t,p);
        else
             // call deleteOnechild(t,p)
        }
}
```

Task 2

Implement a binary search tree in array as shown below and perform insert, delete, traverse, and find methods. Use constructor as needed. The logic for all methods will remain same as you did in linked list-based BST.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	5	8			7	9								12	

Procedure

Algorithm to insert in binary search tree

1. Compare VALUE with root node N of tree
   - If VALUE <N, proceed to the left child of N
   - If VALUE >=N, proceed to the right child of N
2. Repeat step 1. until N=null and assign it a new node with value VALUE

Note: Array implementation formula for left-child is 2P+1 and formula for right child is 2P+2.

```
Class BST<T extends Comparable<T>> {
        T[] tree;

        BST(int size){
        Tree=(T[]) new Comparable[size];
        }
        public void insert(T d){  code here }
        public void find(T d){ code here }
        public void delete(T d) { code here }
        public void traverse(){ code here }
}
```