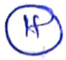
LAB session-05 AVL TREE ROTATIONS

Pre Lab

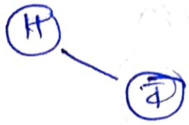
1. given elements

H, I, J, B, A, E, C, F, D, G, K, L

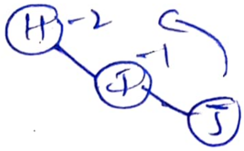
Step 1:



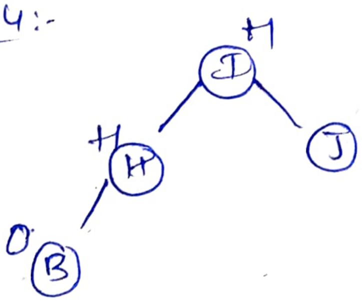
Step 2:



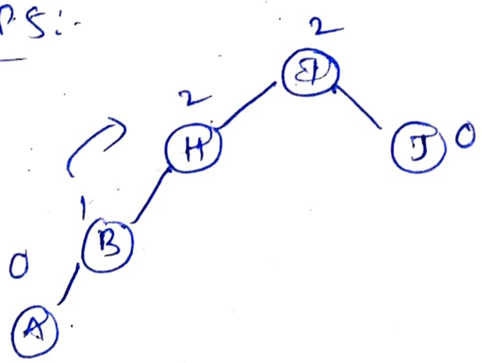
Step 3:-



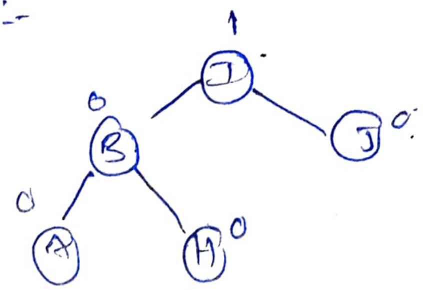
Step 4:-



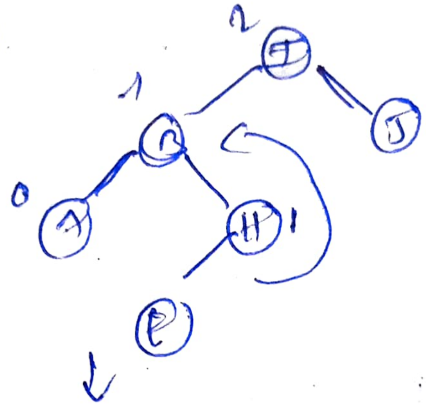
Step 5:-

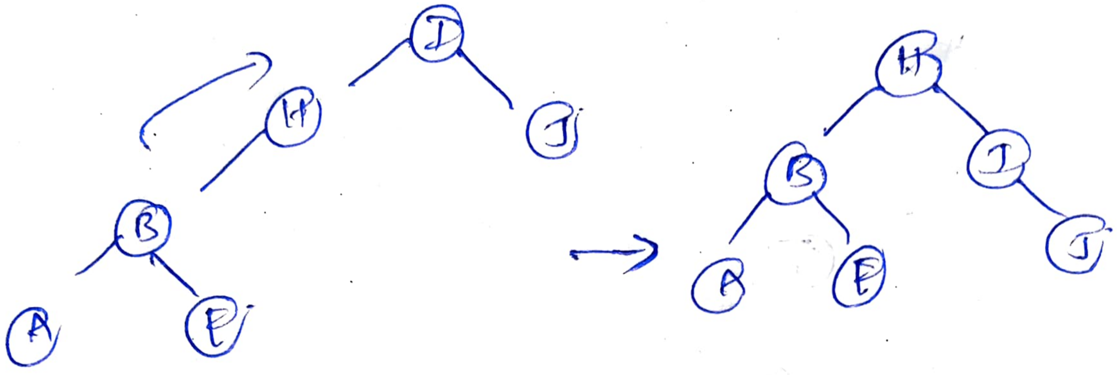


Step 6:-

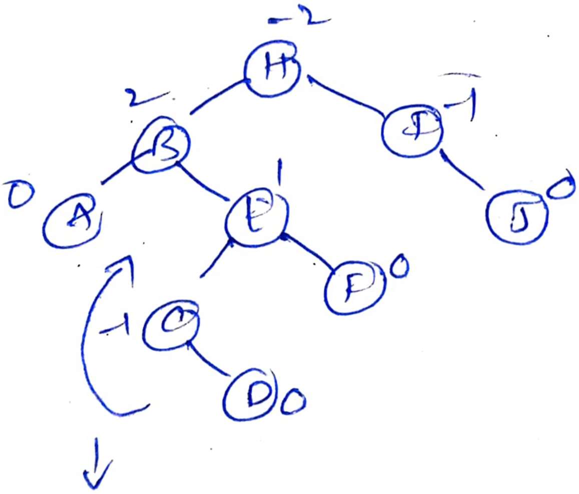


Step-7:

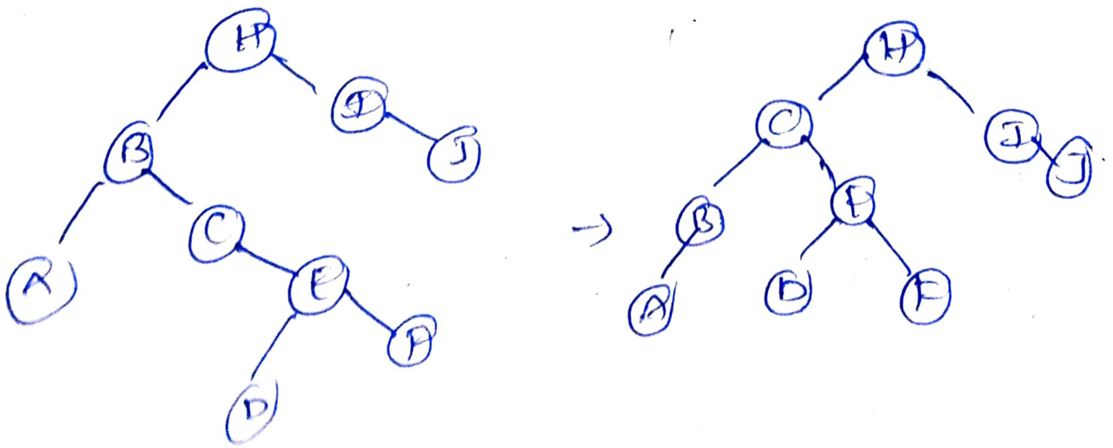




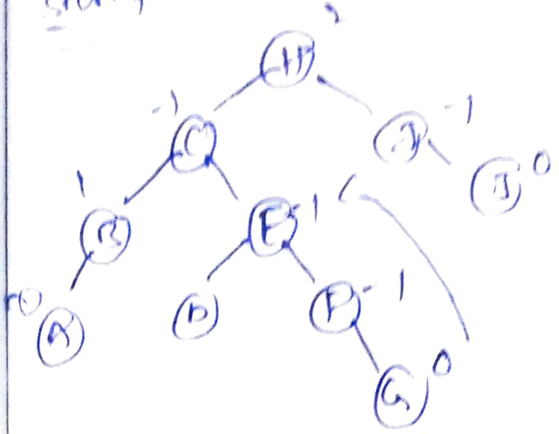
Step-8:

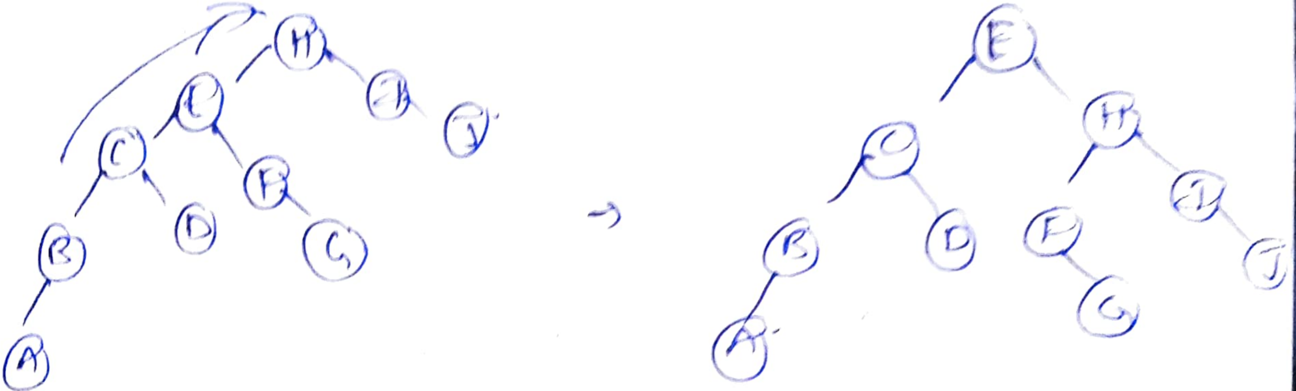


8

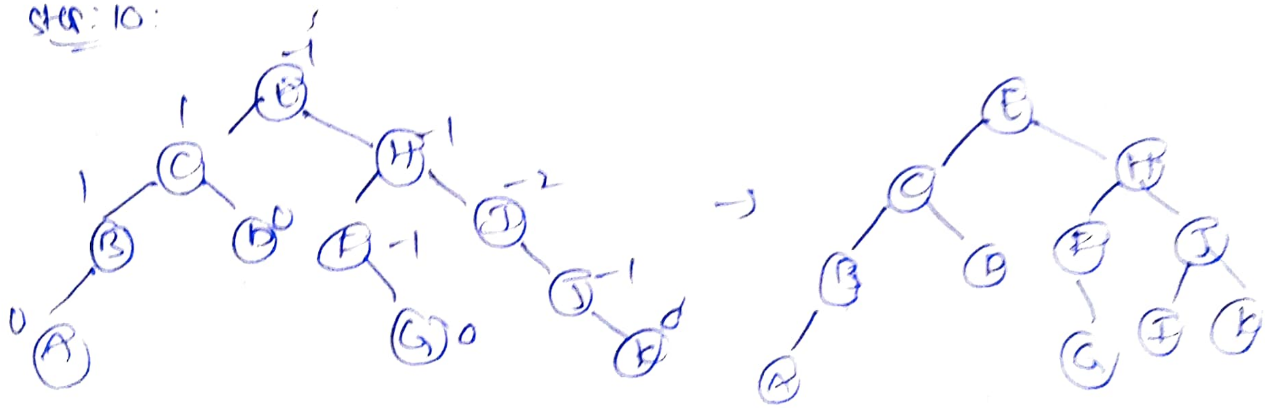


Step. 9

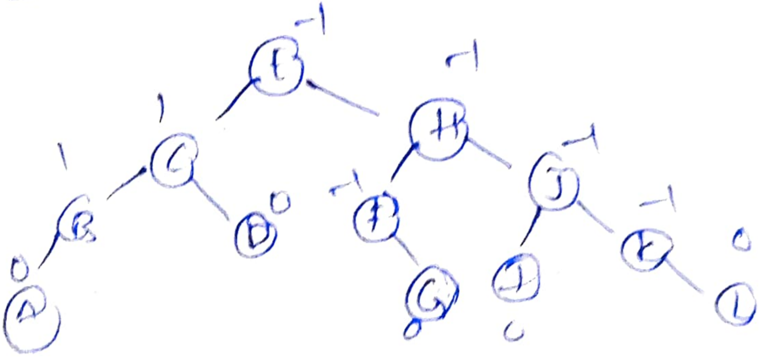




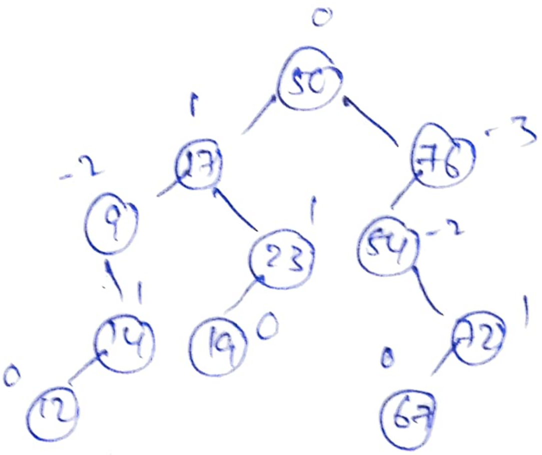
Step: 10:



Step. 11

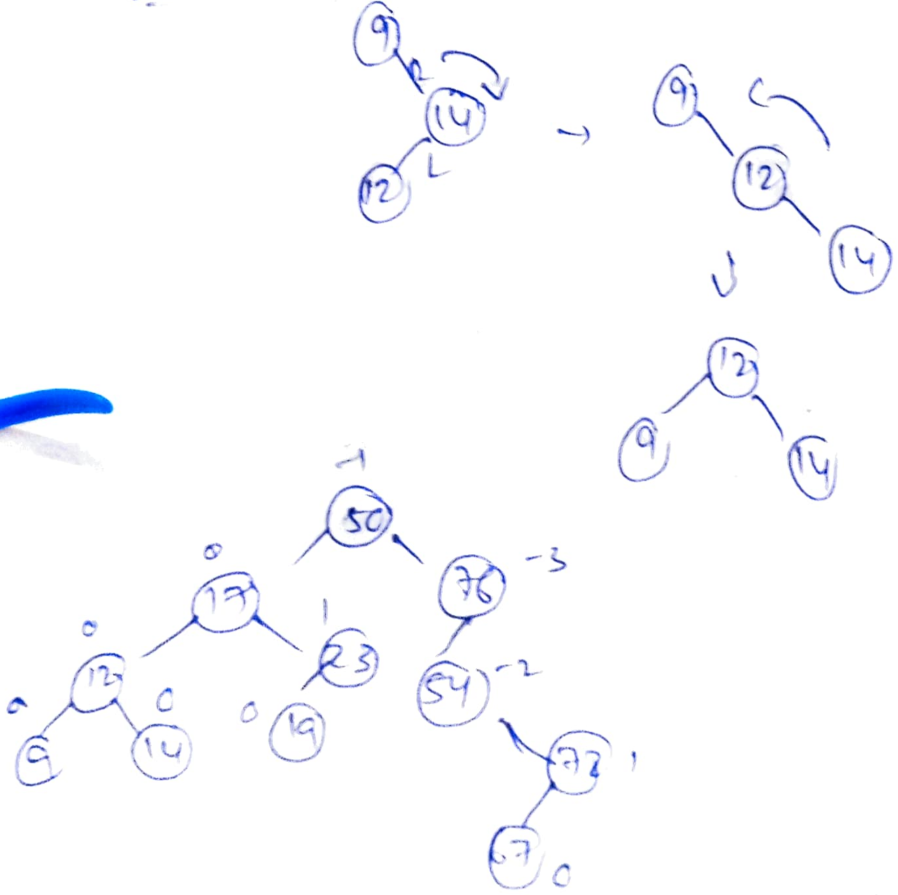


② given unbalanced tree

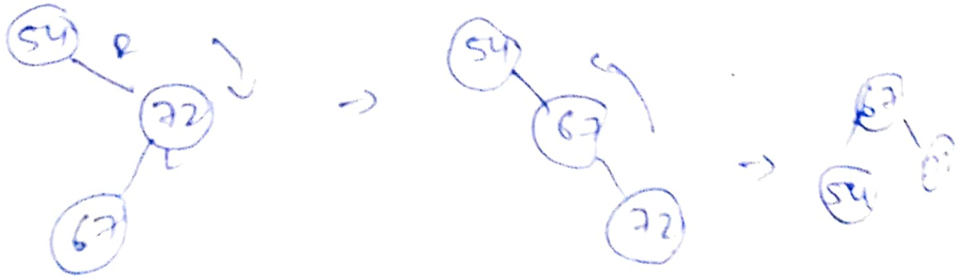


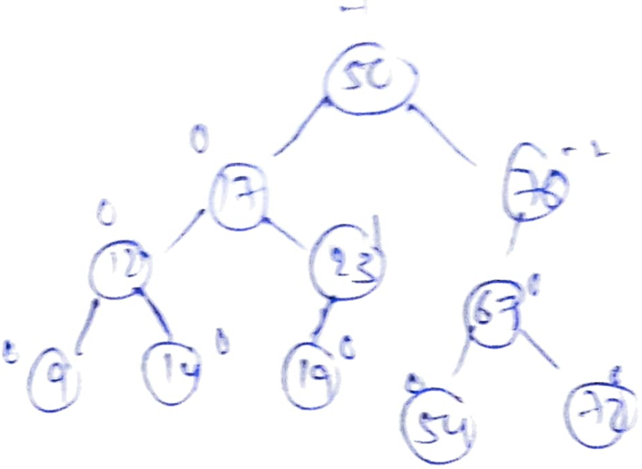
sol:

step1:

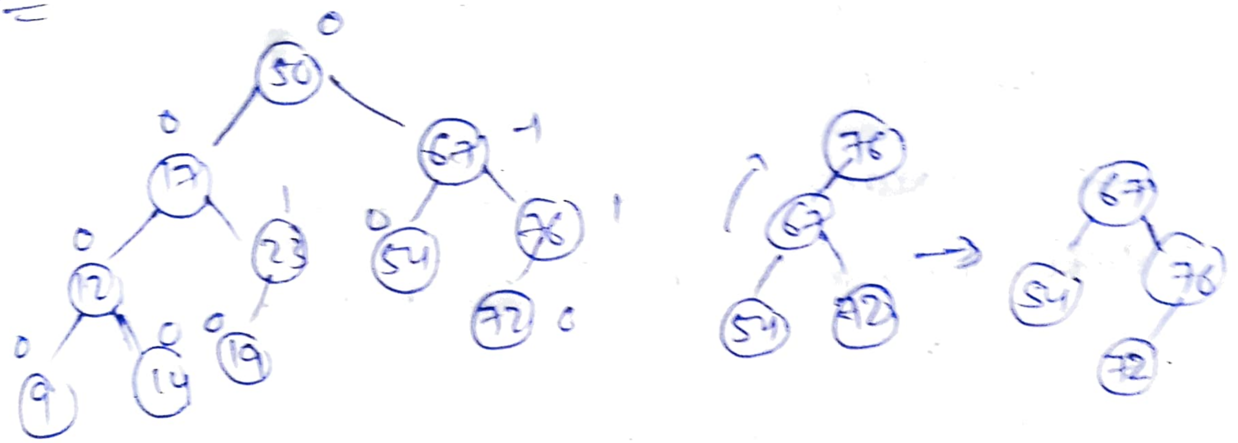


step-3





Step-4



In Lab:-

```
1. package inlab1;
   public class TreeNode {
           int data;
           TreeNode  left;
           TreeNode  right;
           public TreeNode() {
               }
           public TreeNode(int data) {
               this.data = data }
         public TreeNode(int data, TreeNode right
                      TreeNode node left){
```

```
            this.data = data;
            this.left = left;
            this.right = right;
       }
   }
package inlab1;
import Java.util.*;
public class solution{
      List<TreeNode> n= new ArrayList<>();
      public TreeNode balanceBST(TreeNode root) {
              getInorder(root);
           return balance BST(0, n.size()-1);
      }
      public void getInorder(TreeNode node){
            if(node==null)
                 return;
           getInorder(node.left);
           n.add(node);
           getInorder(node.right);
      }

   public TreeNode balanceBST(int left, int right){
              if(left>right)
                    return null;
            int mid = (left+right)/2;
            TreeNode curr = n.get(mid);
```

```
    curr.left = balanceBST( left, mid-1) ;
    curr.right = balanceBST( mid+1 , right ) .

    return curr ;
   }
                                          GAYCl      GACKA
}

package inlab1;
import java.util.*;
public class Demo {

      static List<TreeNode> l = new ArrayList<>( ) ;
     static    Scanner   sc = new   Scanner( System.in) ;
    public  Static  void  main( String [] args) {
            solution   s = new   solution( );
          System.out.println ("enter number of nodes") ;
            int   n = sc.nextInt( );
          system.out.println ("enter nodes");
          for(int i=0 ; i<n ; i++)
                  l.add (addNode( sc.nextInt( )));

          system.out.println( s.balanceBST ( l.get(0)));
   }
  public static TreeNode  addNode( int data) {
          if ( l==null )
          return  new   TreeNode
         else
     }      return  null;
}
```

Postlab

1) root of a binary tree

```
package postlab1;
public class TreeNode{
        int data;
        TreeNode left;
        TreeNode right;
        public TreeNode(){
      }
      public TreeNode(int data){
              this.data= ~~d~~data;
      }
      public TreeNode(int data, TreeNode right,
                                    TreeNode left){
              this.data=data;
              this.left = left;
              this.right = right;
      }
}
package postlab1;
import java.util.*;
public class solution{
        int maxlevel=0;
        int val=0;
      public int findBottomLeftvalue(TreeNode root){
              findval(root,1);
              return val;
   }
```

```
public void findval( TreeNode root, int level) {
        if(root ==null)
              return;
        findval(root.left, level+1);
        if(level >maxlevel) {
              maxlevel= level;
              val= root.data;
        }.
        findval(root.right, level+1);
  }
}
package postLab1;
import java.util.*;
 public class Demo {
      static List<TreeNode   l= new ArrayList<>();
     static Scanner sc= new Scanner(system.in);
     public static void main(string args[]) {
          solution   s= new solution();
         System.out.println("enter no of nodes");
          int n= sc.nextInt();
         system.out.println("enter nodes");
         for(int i=0; i<n; i++)
                 l.add(addNode(sc.nextInt()));
       system.out.println(s.findBottomLeftValue(l.get(0)));
  }
```

```
    public static TreeNode addNode (int data) {
        return new TreeNode(data);
    }
}
```

2. Applications of AVL Trees

1. AVL trees are mostly used for in-memory sorts of sets and dictionaries

2. AVL Trees are also used extensively in data base applications in which insertions and deletions are fewer but there are frequent lookups for data required

3. It is used in applications that require improved searching apart from the database applications