Daily Practice Program Date 19/12/2023

Program 1: Find largest value of each row

```
package ishwarchavan.com;
import java.util.*;

public  class LargestValueInEachRow{   //class created

static class Node {
      int val;
      Node left, right;
};

static void helper(Vector<Integer> res, Node root, int d) {
      if (root == null)
            return;


      if (d == res.size())    // Expand list size
            res.add(root.val);
      else

            res.set(d, Math.max(res.get(d), root.val));

      helper(res, root.left, d + 1);
      helper(res, root.right, d + 1);
}


static Vector<Integer> largestValues(Node root) {    //function to find largest values
      Vector<Integer> res = new Vector<>();
      helper(res, root, 0);
      return res;
}

static Node newNode(int data) {
      Node temp = new Node();
      temp.val = data;      //Assigning to data value
      temp.left = temp.right = null;
      return temp;
}
public static void main(String[] args) {   //main program started

      Node root = null;
      root = newNode(4);
      root.left = newNode(9);
      root.right = newNode(2);
      root.left.left = newNode(3);
      root.left.right = newNode(5);
      root.right.right = newNode(7);

      Vector<Integer> res = largestValues(root);
      for (int i = 0; i < res.size(); i++)    //loop iterating
                  System.out.print(res.get(i)+" ");    //function calling and printing
}
}
```

Program 2: Detect capital

```
package ishwarchavan.com;

public class DetectCapitalNumber {     //class created
      public static void main(String[] args) {
            String word = "Abc";
            System.out.println(detectCapitalUse(word));   //function calling
      }
```

```
    public static boolean detectCapitalUse(String word) {   //function created
       String[] words=word.split(" "); // if a single string contains multiple words
       boolean bool=false;
       for(int i=0; i<words.length; i++){
           bool=detectCapital(words[i]);
           if(!bool) return false;        //checking all the strings are in the form or
not
       }
       return true;
    }
    public static boolean detectCapital(String word){   //function created
        int capital=0;
        int lower=0;
        char b=word.charAt(0);
        if(b>='a' && b<='z')   //checking condition
            return smaller(word);
        for(int i=1; i<word.length(); i++){   //loop iterating
            char a=word.charAt(i);
            if(a>='A' && a<='Z') capital++;
            else lower++;
            if(!(capital==0 || lower==0))   //checking condition
                  return false;
        }
        return true;
    }
    public static boolean smaller(String word){    //for this type of cases mLLLLL
        for(int i=1; i<word.length(); i++){
            char a=word.charAt(i);
            if(a>='A' && a<='Z')
                  return false;    //return false
        }
        return true;
    }
}
```

Program 3: Contigious array

```
package ishwarchavan.com;

import java.util.HashMap;

public class ContinuousArray {   //class created
      public static void main(String[] args) {   //main created
            int[] nums = {0,1,0};
            System.out.println(findMaxLength( nums));   //function calling and printing

      }
    public static int findMaxLength(int[] nums) {   //function created
        int k=0;
        int sum=0;
        int res=0;
        for(int i=0;i<nums.length;i++){   //loop iterating
            if(nums[i]==0)
              nums[i]=-1;
        }
        HashMap<Integer,Integer> h=new HashMap<>();
        h.put(0, -1);
        for(int i=0;i<nums.length;i++){  //loop itarating
            sum+=nums[i];
            if(!h.containsKey(sum))
              h.put(sum,i);
            if(h.containsKey(sum))
              res=Math.max(res,i-h.get(sum)) ;
        }
        return res;   //return res
```

```
    }
}
```

Program 4: Convert to BST to Greater tree

```
package ishwarchavan.com;
import java.io.*;

class Node{    //node created
int data;
Node left, right;

Node(int item){  //A utility function to create a new Binary Tree Node
      data = item;
      left = right = null;
}
}

public class ConvertBSTtoGreaterTree {  //class created

static int sum = 0;
static Node Root;

static void transformTreeUtil(Node root){

      if (root == null)   // Base case
      return;

      transformTreeUtil(root.right);   // Recur for right subtree

      sum = sum + root.data;    // Update sum

      root.data = sum - root.data;   // Store old sum in current node

      transformTreeUtil(root.left);   // Recur for left subtree
}
static void transformTree(Node root){   //A wrapper over transformTreeUtil()

      transformTreeUtil(root);
}

//A utility function to print indorder traversal of a
//binary tree
static void printInorder(Node root){
      if (root == null)
      return;
      printInorder(root.left);
      System.out.print(root.data + " ");
      printInorder(root.right);
}

public static void main (String[] args) {   //main program created

      ConvertBSTtoGreaterTree.Root = new Node(11);
      ConvertBSTtoGreaterTree.Root.left = new Node(2);
      ConvertBSTtoGreaterTree.Root.right = new Node(29);
      ConvertBSTtoGreaterTree.Root.left.left = new Node(1);
      ConvertBSTtoGreaterTree.Root.left.right = new Node(7);
      ConvertBSTtoGreaterTree.Root.right.left = new Node(15);

      System.out.println("Inorder Traversal of given tree");
      printInorder(Root);

      transformTree(Root);    //function calling
```

```
        System.out.println("\n\nInorder Traversal of transformed tree");
        printInorder(Root);
}
}
```