Daily Practice Work:- Date: 24/8/2023

Program:- Find Duplicate Subtree

```
package com.ishwarchavan;

import java.util.HashMap;

public class DuplicateSubtrees {          //class is created

static HashMap<String, Integer> m;        //tree node has data and right and left
pointer
static class Node{
      int data;          //data pointer
      Node left; //right pointer
      Node right;        //left pointer
      Node(int data){
            this.data= data;
            left = null;
            right = null;
      }
}
static String inorder(Node node) {    //functiion is created
      if (node == null)
            return "";

      String str = "(";

            str += inorder(node.left);
            str += Integer.toString(node.data);       //converting integer to string
            str += inorder(node.right);
            str += ")";
            if (m.get(str) != null && m.get(str)==1 )
                  System.out.print( node.data + " ");

            if(m.containsKey(str))        //if true then executed below
                  m.put(str,  m.get(str)+ 1);
            else                          //otherwise
                  m.put(str, 1);
            return str;
}
static void printAllDups(Node root) {           // FUNCTION IS CREATED
      m=new HashMap<>();
      inorder(root);
}
public static void main(String[] args) {        //MAIN program is started
      Node root = null;
      root = new Node(1);
      root.left = new Node(2);
      root.right = new Node(3);
      root.left.left = new Node(4);
      root.right.left = new Node(2);
      root.right.left.left = new Node(4);
      root. right.right = new Node(4);
      printAllDups(root);                 //calling the function
}
}
```

Program:- Insert into a binary search tree

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

class Node{             //class is created for node
    int value;
    Node left,right;          //left and right pointer
```

```
    Node(int value)
    {
        this.value=value;
        left=right=null;
    }
}
class Insertion{
    Node root;          //root of the tree

    Insertion() {        // function created
        root=null;
    }
    public static void preorder(Node ptr) {           ////preorder Traversal of binary
tree

        if(ptr==null)               //  if true then return
        return ;
        System.out.print(ptr.value+" ");
        preorder(ptr.left);         //Calling function
        preorder(ptr.right);
     }
    public void insert(int item) {

        root =insertNode(root,item);  //calling inserNode() method

    }
    public Node insertNode(Node root,int item) {

        if(root==null)     //if root is null create a new Node
        {
            root=new Node(item);
            return root;            //return root
         }
        if(item < root.value)     //if item is less than the current value then
traverse left subtree
            root.left= insertNode(root.left,item);

            else if(item>root.value)   //if item is greater than the current value then
traverse the right subtree
            root.right=insertNode(root.right,item);
            return root;
    }

    public static void main(String[] args) {

        Insertion tree=new Insertion();  //object is created

        tree.insert(30);
        tree.insert(50);
        tree.insert(55);
        tree.insert(45);
        tree.insert(10);
        tree.insert(5);
        tree.insert(15);
        tree.insert(12);
        tree.preorder(tree.root);          //print preorder traversal of binary tree
    }
}
```

Program:- Longest word in dictionary

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

public class LongestWords {         //class is created
      static ArrayList<String> longestWords(String[] dictionary){
```

```
            ArrayList<String> list = new ArrayList<String>();           //object is
created
            int longest_length = 0;       //variable is created
            for(String str : dictionary ) {
                  int length = str.length();
                  if (length>longest_length) {        //loops iterating
                        longest_length = length;            //length variable is
assigned
                        list.clear();
                  }
                  if(length == longest_length) {            //if true then execute the
below statement
                        list.add(str);
                  }
            }
            return list;                  //return list
      }
      public static void main(String[] args) {        //main program is started
            String[] dict = {"cat", "lion","fox","elephant"};
            System.out.println("Origianal dictionary : "+ Arrays.toString(dict));
            System.out.println("Longest word: "+longestWords(dict));
      }
}
```

Program:- Find increasing search order

```
package com.ishwarchavan;

 class IncreasingSearchOrder {            //class is created

       static class node{
            int data;        //data pointer,
            node left;       //left pointer and right pointer is created
            node right;

            node(int data){
                  this.data=data;
                  left = null;
                  right = null;
            }
       }
       static void print(node parent) {         //print function is created
            node curr = parent;
            while(curr != null) {        //conditon is chechked
                  System.out.print(curr.data + " ");
                  curr = curr.right;

            }
       }
       static node prev;
       static void Inorder(node curr) {         //function to perform in order
traversal
            if(curr == null)       //condition is checking
                  return;
            Inorder(curr.left);
            prev.left = null;   //assing value
            prev.right = curr;
            prev = curr;
            Inorder(curr.right);   //calling functiion
       }
       static node flatten(node parent ) {      //function is created
            node dummy = new node(-1);
            prev = dummy;
            Inorder(parent);       //calling function
            prev.left = null;
```

```
            prev.right = null;
            node ret = dummy.right;
            return ret;            //return ret

      }
      public static void main(String[] args) {       //main program is created
            node root = new node(5);
            root.left = new node(3);
            root.right = new node(7);
            root.left.left = new node(2);
            root.left.right = new node(4);
            root.right.left = new node(6);
            root.right.right = new node(8);
            print(flatten(root));         //calling function
      }
}
```

Program:- Check-if-a-binary-tree-is-univalued-or-not

```
package com.ishwarchavan;

import java.util.*;

class Univalued{        //  class is created
      static class Node{
            int data;          //data pointer
            Node left;         //left and right pointer is created
            Node right;
      }
      static Node newNode(int data) {           //new node function is created to
insert new node
            Node temp = new Node();// OBJECT creation
            temp.data= data;
            temp.left = temp.right = null;
            return (temp);
      }
      static boolean isUnivalTree(Node root) {// function is created to check
uninvalued or not
            if(root == null) {
                  return true;
            }
            if(root.left != null && root.data !=root.left.data)         //if condition
satisfied then return false
                  return false;
            if(root.right != null && root.data != root.right.data)            //if
condition is  satisfied then return false
                  return false;
            return isUnivalTree(root.left)&& isUnivalTree(root.right);        //recurse
on left and right subtree
      }
      public static void main(String[] args) {        // main program is started
            Node root = newNode(1);
            root.left = newNode(1);
            root.right= newNode(1);
            root.left.left= newNode(1);
            root.left.right= newNode(1);
            root.right.right= newNode(1);

            if(isUnivalTree(root)) {            //if true then execute below statement
                  System.out.println("Yes");
            }
      }
}
```

Program:- Find Day OF The Year

```
package com.ishwarchavan;
import java.time.LocalDate;

public class DayOfTheYear {          //class is created

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

            LocalDate localDate = LocalDate.now();          //current date or object is
created
            System.out.println("Date: "+localDate);

            int year = localDate.getDayOfYear();            //calling function and
assign in year variable
            System.out.println("Year: "+year);
      }
}
```

Program:- Find Day OF The Year

```
package com.ishwarchavan;

import java.time.DayOfWeek;
import java.time.LocalDate;

public class DayOfTheWeek {                //class is created

      public static void main(String[] args) {              //main program is started
            // Current Date
            LocalDate localDate = LocalDate.now();          //object is created or
current date
            System.out.println("Date: "+localDate);

            DayOfWeek week =  localDate.getDayOfWeek(); // calling function and store
in week variable
            System.out.println("Day of Week: "+week);  // Get Day of Week

      }
}
```

Program:- Check-if-a-word-occurs-as-a-prefix-of-any-word-in-a-sentence

```
package com.ishwarchavan;
import java.util.*;
import java.lang.*;
import java.io.*;
public class PrefixOfWord {          //class is created
      public static int isPrefixOfWord(String sentence, String searchWord) {
//function is created
            String[] words = sentence.split(" ");
            for(int i = 1; i<= words.length; ++i) {          //loops is iterating
                  if(words[i - 1].startsWith(searchWord)) {
                        return i;
                  }
            }
            return -1;              //return -1
      }
      public static void main(String[] args) {  //main program is started
            String sentence = "i love my India";
            String searchWord = "India";
            System.out.println(isPrefixOfWord(sentence, searchWord));   //calling and
print the value
      }
}
```

Program:- Minimum-insertions-to-balance-a-parentheses-string

```
package com.ishwarchavan;

public class MinimumNumberOfParethesis {        //class is created

static int minParentheses(String p) {           //function is created to find minimum
number

      int bal = 0;            //variable initiolized and decalared
      int ans = 0;
      for(int i = 0; i < p.length(); ++i) {           //loops is iterating

            bal +=p.charAt(i) == '(' ? 1 : -1;  //condition operator checking
            if(bal == -1) {
                  ans +=1;
                  bal += 1;
            }

      }
      return bal+ans;         //return the value
}
      public static void main(String[] args) {        //main program is started
            String p = "())";
            System.out.println(minParentheses(p));
      }
}
```

Program:- Convert-1d-array-into-2d-array

```
package com.ishwarchavan;

public class Conver1dArrayInto2dArray {         //class is created

      static void arr(int[] a, int row, int col){  //arr function is created
            int i = 0;              //initialize the with 0
            int [][]two = new int[row][col];          //OBJECT IS created

            for(int y=0; y<row; y++) {          //loops1 iterating

                  for(int x=0; x<col; x++) {          //loops2 iterating
                        two[y] [x] =  a[i];
                        i++;        //incrementing value
                        System.out.print(two[y][x] + " ");
                  }
                  System.out.println();         //printing empty value
            }
      }

      public static void main(String[] args) {        //main program is started
            int[] a = new int[] {1,2,3,4};
            arr(a,2,2);             //calling function with passing parameter
      }
}
```

Program:- Vowels-of-all-substrings

```
package com.ishwarchavan;

public class VowelOfAllSubstring {   //class is created

      public static void main(String[] args) {        //main program is started
            String word="abc";
            System.out.println(countVowels(word));          //calling and printing
value
      }
```

```
                public static long countVowels(String word) {           //function is
created

                    long res = 0, prev = 0;                 //initialize and declaring
variables
                    for(int i=0; i<word.length(); i++) {          //loops iterating
                        char c = word.charAt(i);
                        if(c == 'a' || c == 'e' || c == 'i' || c == 'o' || c ==
'u')//condition to check vowels
                            prev += i + 1;
                        res += prev;
                    }

                    return res;           //return res variables value value
                }
      }
```

Program:- Find Common Factors

```
package com.ishwarchavan;

public class NumberOfCommonFactor {       //class is created

      static int gcd(int a, int b) {            //function is created to find gcd of
two numbers
      if(a == 0)              //if true then return b
            return b;

      return gcd(b % a,a);    //otherwise return this
}
      static int commDiv(int a, int b) {        //function is created to find common
divisors numbers
            int n = gcd(a,b);             //Calling and storing

            int result = 0;         //count divisors of n
            for(int i= 1; i <= Math.sqrt(n); i++) {

                  if(n % i == 0) {        //if i is factor of n

                        if(n / i == i)          //checking divisor are equal or not
                              result +=1;
                        else
                              result +=2;
                  }
                  }
            return result;          // return the value of result
            }
      public static void main(String[] args) {        //main program is started
            int a= 12, b=24;
            System.out.println(commDiv(a,b));  //calling and printing
      }
}
```

Program:- Find Closest Number

```
package com.ishwarchavan;

class FindClosestNum {        //class is created

      public static void main(String[] args) {  //main program is started
            int[] nums= {-4,-2,1,4,8};
            System.out.println(findClosestNumber(nums));  //calling and printing value
      }
    public static int findClosestNumber(int[] nums) {   //main program is started
        int min = Integer.MAX_VALUE, closest_num = 0;       //storing max integer value
        for(int n : nums) {
```

```
            if(min > Math.abs(n)) {       //if condition is true then execute below
statement
                min = Math.abs(n);
                closest_num = n;
            } else if(min == Math.abs(n) && closest_num < n) {                // if
this condition is true then assign n value to the closest_num variable
                closest_num = n;
            }
        }
        return closest_num;         //return the closest_num value
    }
}
```