Daily Work Practice:- Date – 11/8/2023

Program:- Merge two arrays

```
package com.ishwarchavan;

import java.util.*;


public class MergeTwoArrays {       //class created

      public static void main(String[] args) {        //main program started
            int[]a = {1,2,3,4,5,};        //initializing and declaring arrays
            int[]b = {6,7,8};
            int ci = 0;

            int[]c = new int[a.length + b.length];    //creating new instance here
            for(int i =0; i<a.length; i++) {          //for loop iteraring
                  c[ci]=a[i];
                  ci++;             // if satisfied the condition then increment it
            }
            for (int i=0;i<b.length; i++) {           //iterating loop here
                  c[ci]= b[i];
                  ci++;                                     // if satisfied the
condition then increment it
            }
            System.out.println(Arrays.toString(c));

      }

}
```

Program:- Maximum Width Of Binary Tree

```
package com.ishwarchavan;
import java.util.LinkedList;
import java.util.Queue;
public class MaximumWidthOfBinaryTree {
       static class TreeNode{       //a binary tree node has two pointer
            int data;
            TreeNode left, right;

            public TreeNode(int data) {this.data=data;}

       }
       static int maxwidth(TreeNode root) {           //maxwidth function is created to
find the maximum width of the tree
             if(root == null)       //if satisfied the condition then return 0
                   return 0;

             int maxwidth = 0;            //initialize the result in this variable

            Queue<TreeNode> q= new LinkedList<>();          //instance q is created
here for traversal nodes
            q.add(root);

            while(!q.isEmpty()) {         //condition is checking
                  int count = q.size();  //store the size of queue

                  maxwidth = Math.max(maxwidth, count);  //updating the maximum node
count value

                  while(count-- > 0) {          /dequeue an node from queue
```

```
                        TreeNode temp=    q.remove();

                        if(temp.left !=null) {        //if it true then enqueue left
node
                              q.add(temp.left);
                        }
                        if(temp.right !=null) {     //if it true then enqueue right
node
                              q.add(temp.right);
                        }
                  }
            }
             return maxwidth;       // return the maxwidth
       }
         public static void main(String[] args) {           //main program started here
               TreeNode root = new TreeNode(1);
            root.left = new TreeNode(3);
            root.right =new TreeNode(2);
            root.left.left = new TreeNode(5);
            root.left.right =new TreeNode(3);
            root.right.right=new TreeNode(8);
            root.right.right.left=new TreeNode(9);

            System.out.println("Maximum width=" + maxwidth(root));


      }
}
```