CS162 ASSIGNMENT 11

NAME:

ARCHIT AGRAWAL

ROLL NO. :

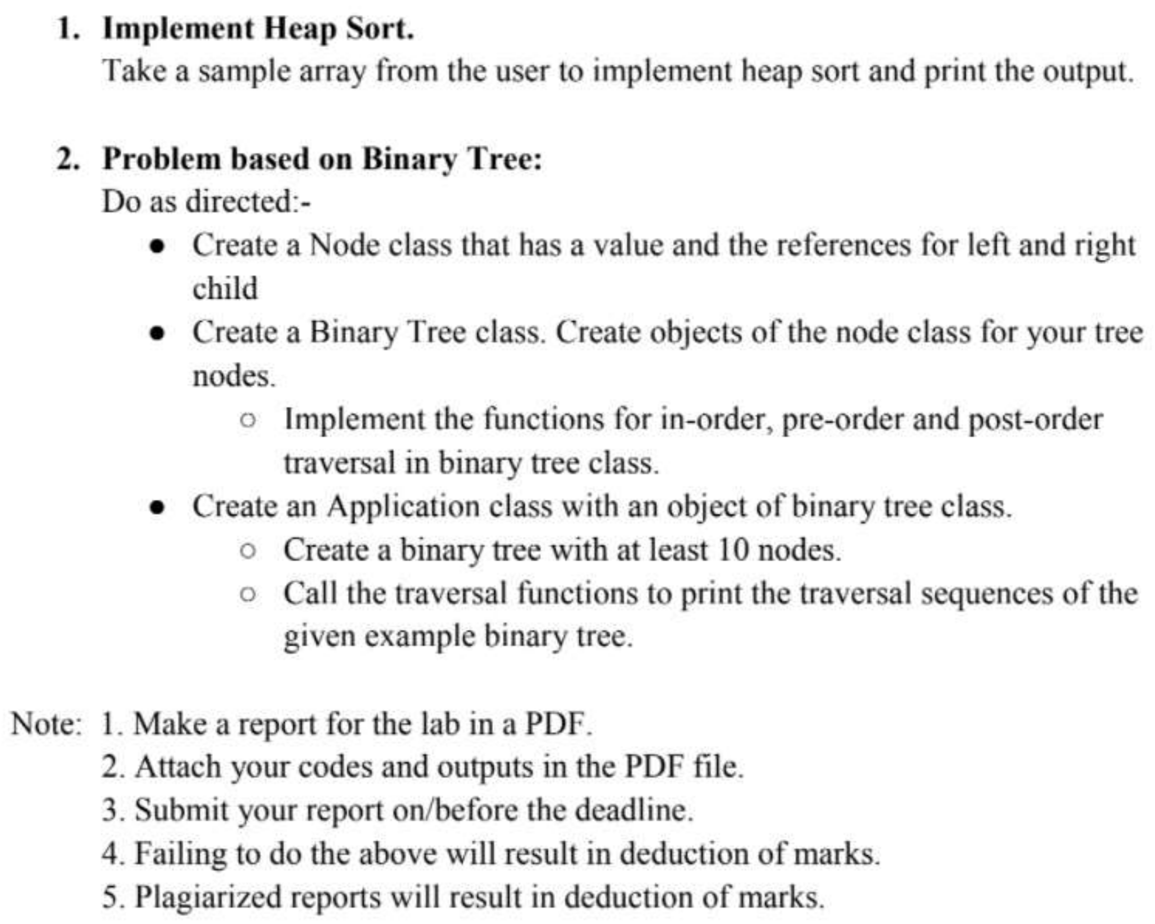
202052307

SECTION:

A

ARCHIT AGRAWAL 202052307

Question



1. **Implement Heap Sort.**
   Take a sample array from the user to implement heap sort and print the output.

2. **Problem based on Binary Tree:**
   Do as directed:-
   - Create a Node class that has a value and the references for left and right child
   - Create a Binary Tree class. Create objects of the node class for your tree nodes.
     - Implement the functions for in-order, pre-order and post-order traversal in binary tree class.
   - Create an Application class with an object of binary tree class.
     - Create a binary tree with at least 10 nodes.
     - Call the traversal functions to print the traversal sequences of the given example binary tree.

Note: 1. Make a report for the lab in a PDF.
2. Attach your codes and outputs in the PDF file.
3. Submit your report on/before the deadline.
4. Failing to do the above will result in deduction of marks.
5. Plagiarized reports will result in deduction of marks.

CODE

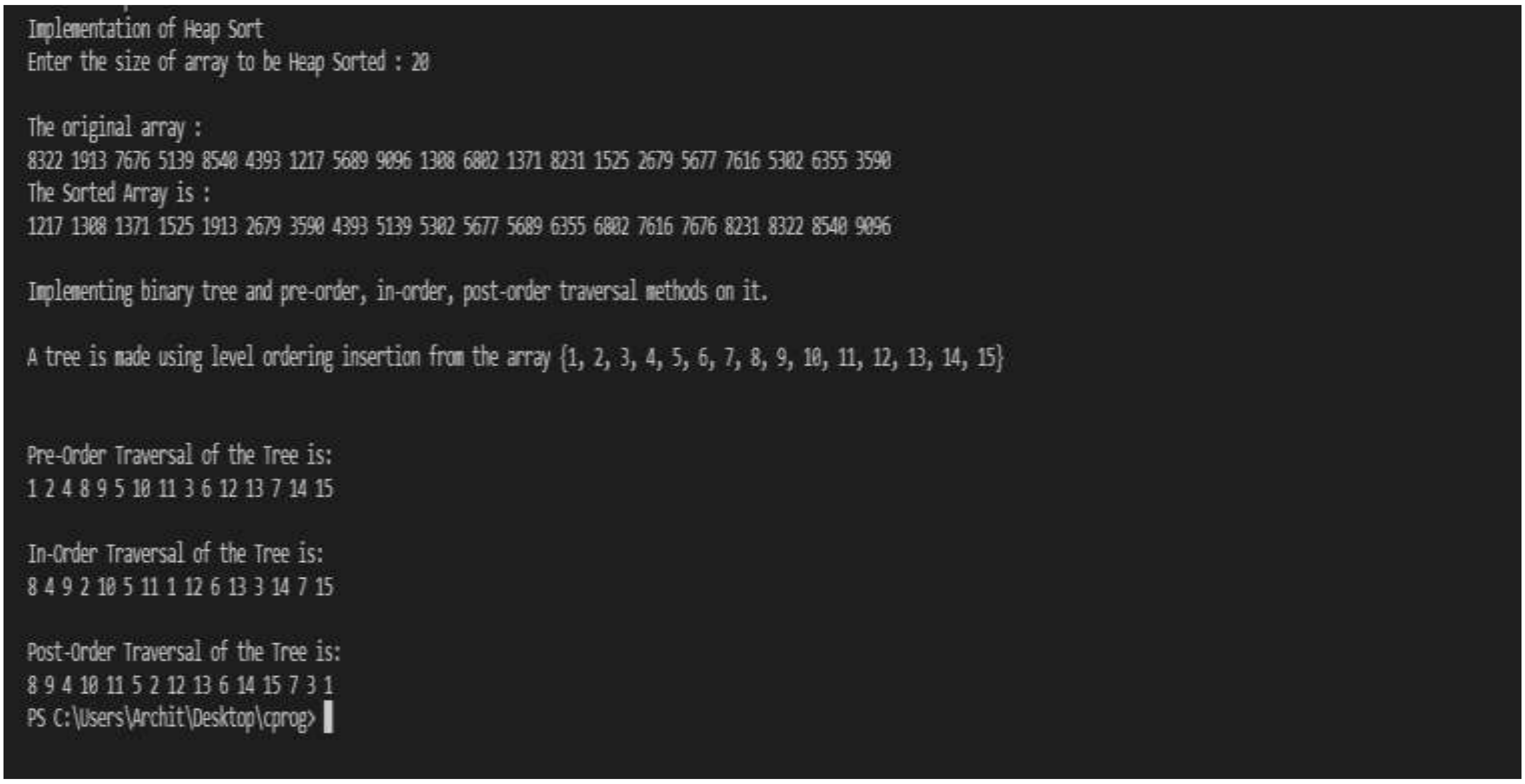
```java
//Archit Agrawal
//202052307
//This code contains the code for Heap sort
//and implementation of Binary tree with pre-order, in-order, post-order
//traversal methods

import java.util.*;

class HeapSort{

    void heapify(Integer []arr, int n, int index){
        int largest = index;
        int l = 2 * index + 1;
        int r = 2 * index + 2;

        if(l < n && arr[l] > arr[largest]){
            largest = l;
        }

        if(r < n && arr[r] > arr[largest]){
            largest = r;
        }

        if(largest != index){
            int temp = arr[largest];
            arr[largest] = arr[index];
            arr[index] = temp;

            heapify(arr, n, largest);
        }
    }

    void sort(Integer []arr){

        int n = arr.length;

        for(int i = n/2 - 1; i >= 0; i--){
            heapify(arr, n, i);
        }

        for(int i = n - 1; i >= 0; i--){
            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;

            heapify(arr, i, 0);
        }
    }

}

class Node<T>{
    T data;
    Node<T>left;
    Node<T> right;

    public Node(T key){
        this.data = key;
        this.left = null;
        this.right = null;
    }
}

class BinaryTree<T>{

    Node<T> root;

    public BinaryTree(){
        root = null;
    }

    public BinaryTree(T key){
        root = new Node(key);
    }

    public Node<T> constructLevelOrder(T []arr, Node root, int index){
        if(index < arr.length){
            Node<T> temp =  new Node<T>(arr[index]);
            root = temp;

            root.left = constructLevelOrder(arr, root.left, 2 * index + 1);
            root.right = constructLevelOrder(arr, root.right, 2 * index + 2);
        }

        return root;
    }

    public void preOrderTraversal(Node node){
        if(node == null) return;
        //pre-order is node-left-right(NLR) traversal
        System.out.print(node.data + " ");
        preOrderTraversal(node.left);
        preOrderTraversal(node.right);
    }

    public void inOrderTraversal(Node node){
        if(node == null) return;
        //in-order is left-node-right(LNR) traversal
        inOrderTraversal(node.left);
        System.out.print(node.data + " ");
        inOrderTraversal(node.right);
    }

    public void postOrderTraversal(Node node){
        if(node == null) return;
        //post-order is left-right-node(LRN) traversal
        postOrderTraversal(node.left);
        postOrderTraversal(node.right);
        System.out.print(node.data + " ");
    }

    public void preOrderTraversal(){
        preOrderTraversal(root);
    }

    public void inOrderTraversal(){
        inOrderTraversal(root);
    }

    public void postOrderTraversal(){
        postOrderTraversal(root);
    }
}

public class Application{
    public static void main(String []args){

        //IMPLEMENTING HEAP SORT
        Scanner sc = new Scanner(System.in);

        System.out.println("Implementation of Heap Sort");
        System.out.print("Enter the size of array to be Heap Sorted : ");
        int n = sc.nextInt();

        System.out.println();
        System.out.println("The original array :");

        Integer[] a = new Integer[n];
        Random rand = new Random();
```

```java
        for (int i = 0; i < n; i++) {
            a[i] = rand.nextInt(9000) + 1000;
        }

        for(Integer i: a){
            System.out.print(i + " ");
        }

        HeapSort hs = new HeapSort();
        hs.sort(a);

        System.out.println();
        System.out.println("The Sorted Array is :");

        for(Integer i : a){
            System.out.print(i + " ");
        }

        System.out.println();
        System.out.println();
        System.out.println("Implementing binary tree and pre-order, in-
order, post-order traversal methods on it.");
        //IMPLEMENTION BINARY TREE AND PRE-ORDER, IN-ORDER, POST-
ORDER TRAVERSALS
        BinaryTree<Integer> tree = new BinaryTree<Integer>();
        System.out.println();
        System.out.println("A tree is made using level ordering insertion from
 the array {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}");
        //this is a BinaryTree object made using the constructLevelOrder metho
d
        Integer [] arr = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};
        tree.root = tree.constructLevelOrder(arr, tree.root, 0);

        System.out.println();
        System.out.println();
        System.out.println("Pre-Order Traversal of the Tree is:");
        tree.preOrderTraversal();
        System.out.println();
        System.out.println();
        System.out.println("In-Order Traversal of the Tree is:");
        tree.inOrderTraversal();
        System.out.println();
        System.out.println();
        System.out.println("Post-Order Traversal of the Tree is:");
        tree.postOrderTraversal();

    }
```

```
}
```

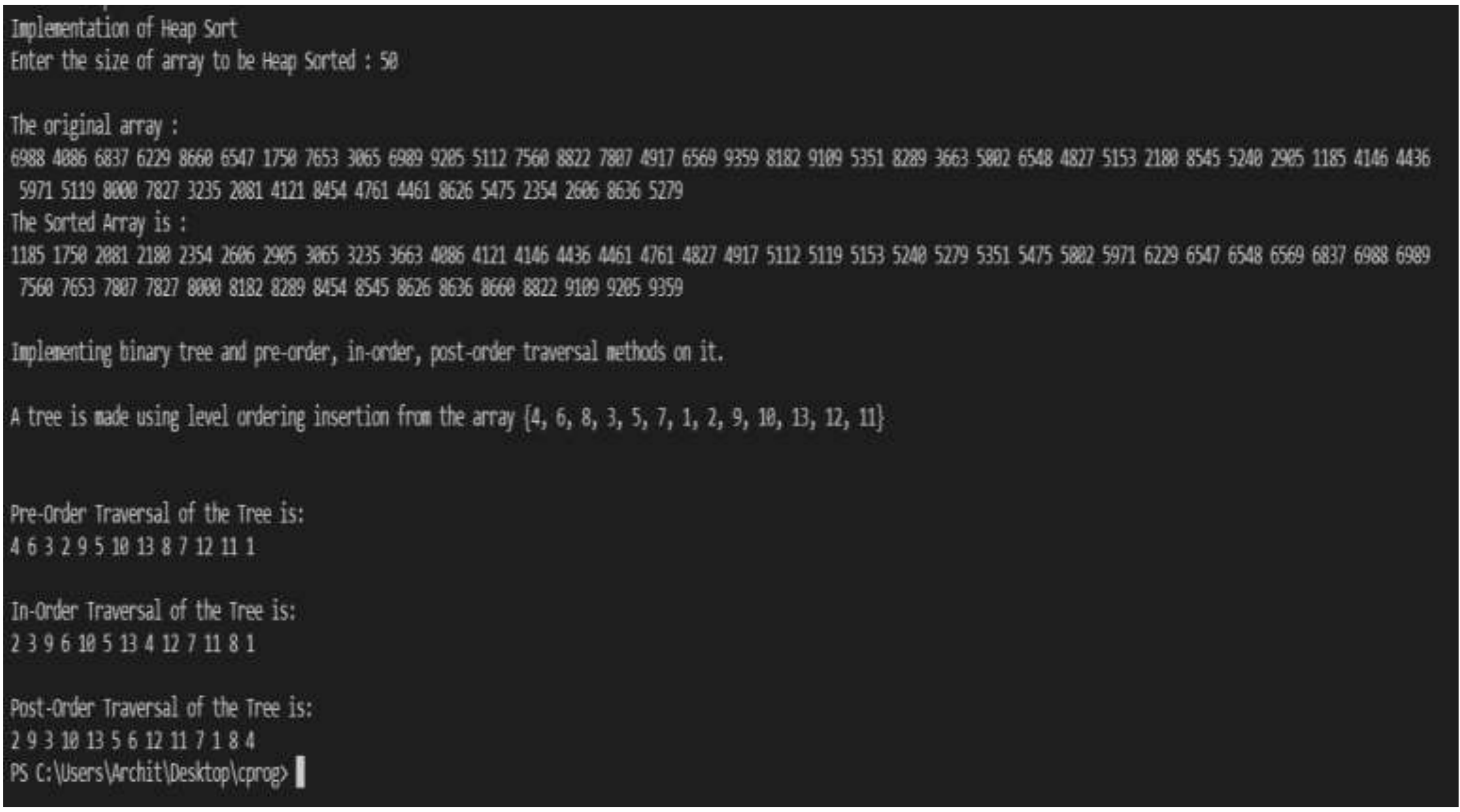
OUTPUT

Output 1:



```
Implementation of Heap Sort
Enter the size of array to be Heap Sorted : 20

The original array :
8322 1913 7676 5139 8540 4393 1217 5689 9096 1308 6802 1371 8231 1525 2679 5677 7616 5302 6355 3590
The Sorted Array is :
1217 1308 1371 1525 1913 2679 3590 4393 5139 5302 5677 5689 6355 6802 7616 7676 8231 8322 8540 9096

Implementing binary tree and pre-order, in-order, post-order traversal methods on it.

A tree is made using level ordering insertion from the array {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}


Pre-Order Traversal of the Tree is:
1 2 4 8 9 5 10 11 3 6 12 13 7 14 15

In-Order Traversal of the Tree is:
8 4 9 2 10 5 11 1 12 6 13 3 14 7 15

Post-Order Traversal of the Tree is:
8 9 4 10 11 5 2 12 13 6 14 15 7 3 1
PS C:\Users\Archit\Desktop\cprog>
```

Output 2:



```
Implementation of Heap Sort
Enter the size of array to be Heap Sorted : 50

The original array :
6988 4086 6837 6229 8660 6547 1750 7653 3065 6989 9205 5112 7560 8822 7807 4917 6569 9359 8182 9109 5351 8289 3663 5802 6548 4827 5153 2180 8545 5240 2905 1185 4146 4436
 5971 5119 8000 7827 3235 2081 4121 8454 4761 4461 8626 5475 2354 2606 8636 5279
The Sorted Array is :
1185 1750 2081 2180 2354 2606 2905 3065 3235 3663 4086 4121 4146 4436 4461 4761 4827 4917 5112 5119 5153 5240 5279 5351 5475 5802 5971 6229 6547 6548 6569 6837 6988 6989
 7560 7653 7807 7827 8000 8182 8289 8454 8545 8626 8636 8660 8822 9109 9205 9359

Implementing binary tree and pre-order, in-order, post-order traversal methods on it.

A tree is made using level ordering insertion from the array {4, 6, 8, 3, 5, 7, 1, 2, 9, 10, 13, 12, 11}


Pre-Order Traversal of the Tree is:
4 6 3 2 9 5 10 13 8 7 12 11 1

In-Order Traversal of the Tree is:
2 3 9 6 10 5 13 4 12 7 11 8 1

Post-Order Traversal of the Tree is:
2 9 3 10 13 5 6 12 11 7 1 8 4
PS C:\Users\Archit\Desktop\cprog> █
```