DAILY ONLINE ACTIVITIES SUMMARY

Date:	**27-05-2020**	**Name:**	**Anvitha Poojary**
Sem & Sec	**6A**	**USN:**	**4AL17CS008**

Online Test Summary

Subject	**SSCD**		
Max. Marks	**30**	**Score**	**14**

Certification Course Summary

Course	**Introduction to Ethical Hacking**		
Certificate Provider	**greatlearning**	**Duration**	**6hr**

Coding Challenges

Problem Statement:

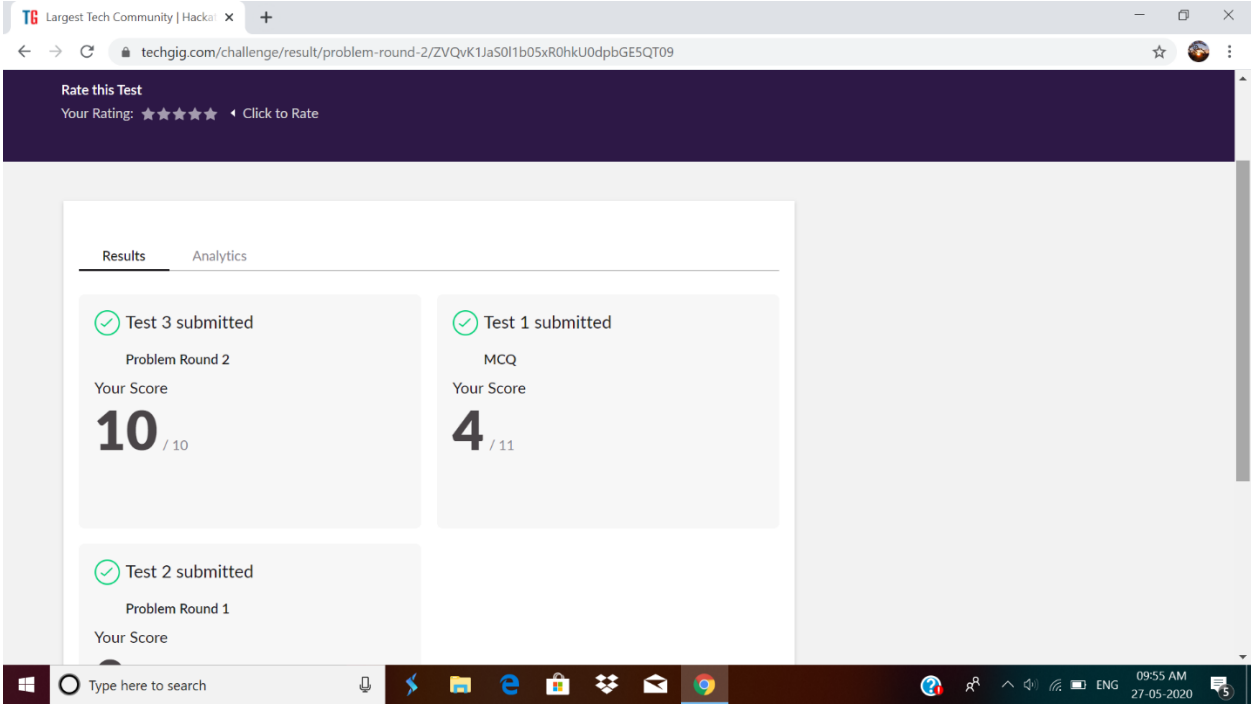
1. Given an array arr[] of the positive integers of size N, the task is to find the largest element on the left side of each index which is smaller than the element present at that index. Note: If no such element is found then print -1.
2. Write a Java program to implement Binary Tree using the Linked List
3. Bubble sort

Status: completed

Uploaded the report in Github	**Yes**
If yes Repository name	https://github.com/anvithapo99/Daily-Report
Uploaded the report in slack	**Yes**

Online test details:

Subject: SSCD



Certification course details:

<u>Introduction to Ethical Hacking:</u>

Today I have studied following topics:

- What is hacking
- Computer security threats
- Goals of ethical hacking
- Skills required by ethical hackers

2:45 50%

greatlearning
Learning for Life

Applications-Demonstration
50m

Ethical Hacking on Mobile Platforms-Demonstration
34m

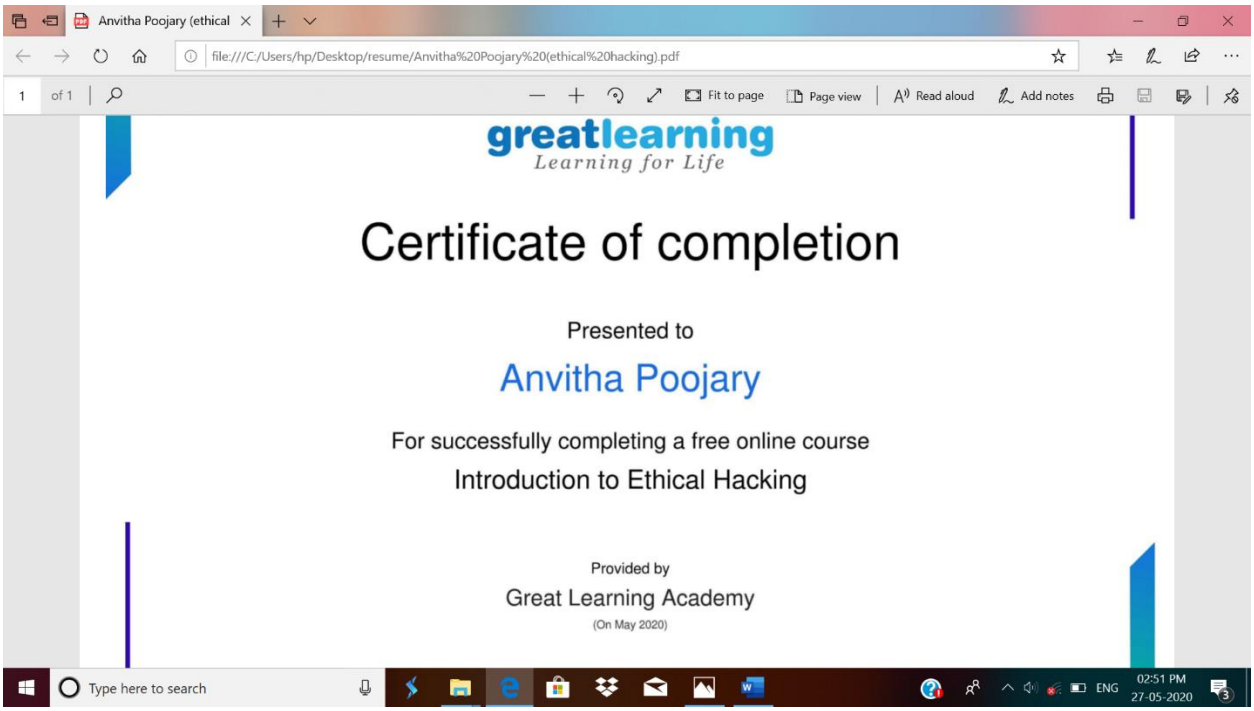
What is Ethical Hacking
50m

Quiz

Ethical Hacking - Quiz
Your Score: 10/10

Claim Your Course Certificate

Claim your course certificate
Your Score: 0/1



greatlearning
Learning for Life

Certificate of completion

Presented to

Anvitha Poojary

For successfully completing a free online course

Introduction to Ethical Hacking

Provided by

Great Learning Academy

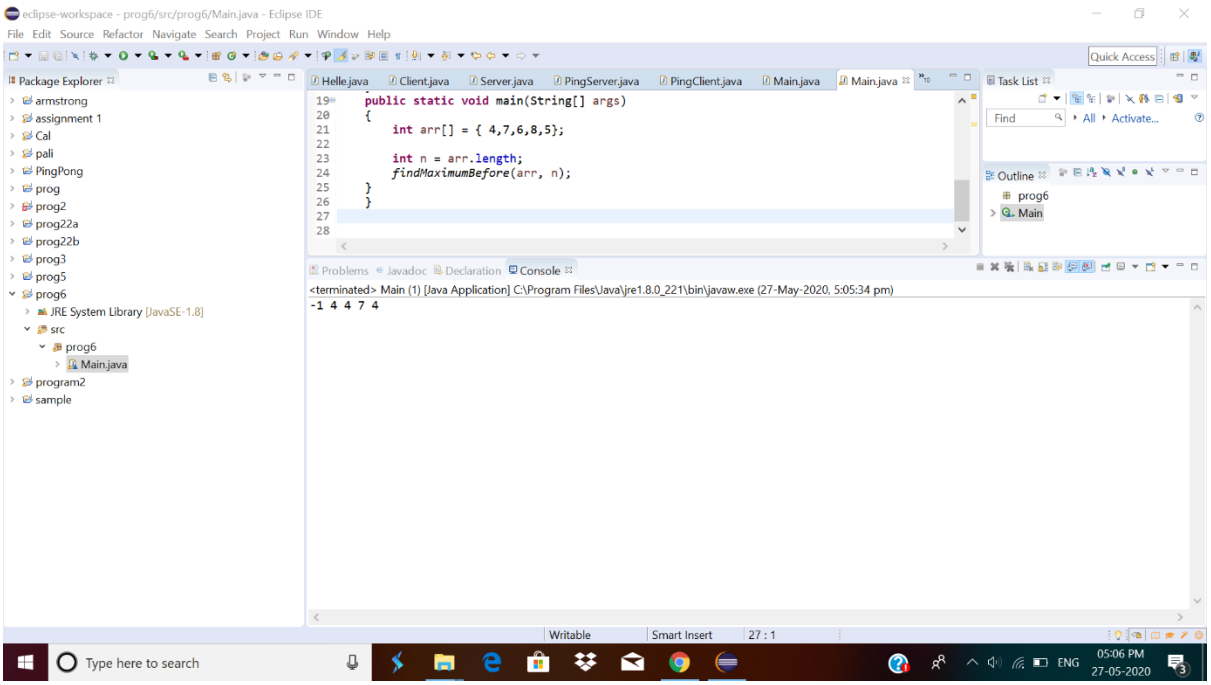
(On May 2020)

1. Given an array arr[] of the positive integers of size N, the task is to find the largest element on the left side of each index which is smaller than the element present at that index. Note: If no such element is found then print -1.

```
import java.util.*;

class Main{

static void findMaximumBefore(int arr[], int n){

    for (int i = 0; i < n; i++) {

     int currAns = -1;

            for (int j = i - 1; j >= 0; j--) {

                    if (arr[j] > currAns &&

                            arr[j] < arr[i]) {

                            currAns = arr[j];

                    }

            }

            System.out.print(currAns+ " ");

    }
```

```
}

public static void main(String[] args)

{

    int arr[] = { 4,7,6,8,5};


    int n = arr.length;

    findMaximumBefore(arr, n);

}

}
```

Output:



2. Write a Java program to implement Binary Tree using the Linked List

In this program, we need to create the binary tree by inserting nodes and displaying nodes in in-order fashion. A typical binary tree can be represented as follows:

Java program to implement Binary Tree using the Linked List
In the binary tree, each node can have at most two children. Each node can have zero, one or two children. Each node in the binary tree contains the following information:

Java program to implement Binary Tree using the Linked List
Data that represents value stored in the node.

Left that represents the pointer to the left child.

Right that represents the pointer to the right child.

Algorithm
Define Node class which has three attributes namely: data left and right. Here, left represents the left child of the node and right represents the right child of the node.

When a node is created, data will pass to data attribute of the node and both left and right will be set to null.
Define another class which has an attribute root.
Root represents the root node of the tree and initialize it to null.
a. insert() will add a new node to the tree:

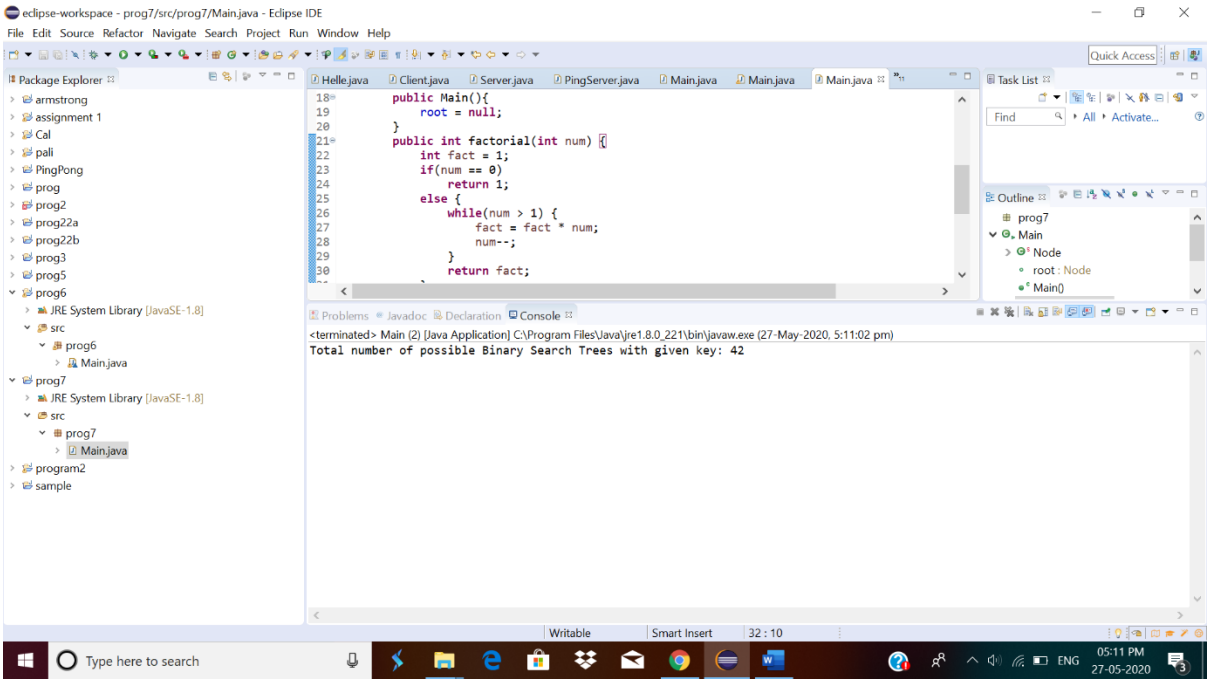
It checks whether the root is null, which means the tree is empty. It will add the new node as root.
Else, it will add root to the queue.
The variable node represents the current node.
First, it checks whether a node has a left and right child. If yes, it will add both nodes to queue.
If the left child is not present, it will add the new node as the left child.
If the left is present, then it will add the new node as the right child.
a. Inorder() will display nodes of the tree in inorder fashion.

It traverses the entire tree then prints out left child followed by root then followed by the right child.

```
public class BinarySearchTree {

   public static class Node{

      int data;

      Node left;

      Node right;
```

```
    public Node(int data){
        this.data = data;
        this.left = null;
        this.right = null;
        }
    }
public Node root;


public BinarySearchTree(){
    root = null;
}
public int factorial(int num) {
    int fact = 1;
    if(num == 0)
        return 1;
    else {
        while(num > 1) {
            fact = fact * num;
            num--;
        }
        return fact;
    }
public int numOfBST(int key) {
    int catalanNumber = factorial(2 * key)/(factorial(key + 1) * factorial(key));
    return catalanNumber;
```

```
    }

 public static void main(String[] args) {

  BinarySearchTree bt = new BinarySearchTree();

 System.out.println("Total number of possible Binary Search Trees with given key: " + bt.numOfBST(5));

      }

}
```



3.In Bubble sort, each pass consists of comparison each element in the file with its successor (i.e. x[i] with x[i+1]) and interchanging two elements if they are not in the proper order. The array may be sorted in any pass. If the array is sorted, then remaining passes should be skipped off. Write a C Program to sort an array of integers in ascending order and display the sorted array and Number of passes performed for sorting.

```
#include <stdio.h>
```

```
int main()
{
  int array[100], n, c, d, swap;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  for (c = 0 ; c < n - 1; c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if (array[d] > array[d+1]) /* For decreasing order use < */
      {
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }
```

```
  printf("Sorted list in ascending order:\n");


  for (c = 0; c < n; c++)

    printf("%d\n", array[c]);


  return 0;

}
```

Output: