CS211 Data Structures and Algorithms

Lab Exam I

Duration: 1.5 Hours

Total Questions: 3

Submission Mode: Turnitn.com

DON’T TURN THIS PAGE UNTIL INSTRUCTED.

1. **You are required to write a function that takes a parameter 'n'. The function should take 'n' number inputs from the users. Then program should ask a number to search and print "Found" number if found, otherwise just print "Not found". The searching should be done in O(log n) times.**

Following is the sample output for n = 4

```
Enter Number 1: 2
Enter Number 2: 7
Enter Number 3: 1
Enter Number 4: -5

Enter number to search: 7
Found
```

Following is the sample output for n = 3

```
Enter Number 1: 12
Enter Number 2: 17
Enter Number 3: 1

Enter number to search: 7
Not Found
```

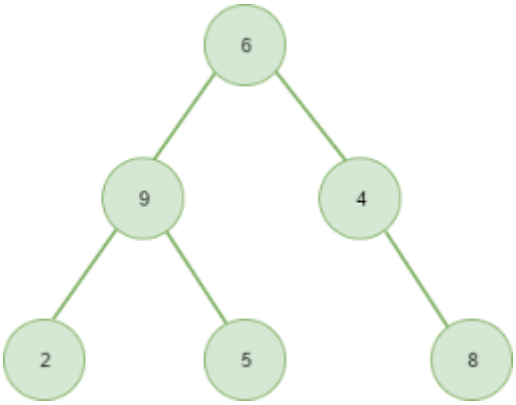
Signature:

void Question1()

2. **You have to write a function to height of the given node. The height of a node is the length of the path form the node to the deepest leaf. The pseudo-code of the recursive function of height is given bellow:**

```
Height: Given n = node
        If n is not a leaf
                Height = 1 + Max ( Height (n -> left, n-> right)
```

As you can see the height of node 6 in the following binary tree is 2



You can use following class to represent a Binary Node

```
class BNode
{
        public BNode left;
        public BNode right;
        public int Data;
}
```

The following is the basic implementation of Binary Search Tree which you can use for your solution. You have to write your function in BTree class.

```
public class BinarySearchTree
    {
        private BNode root;

        public BinarySearchTree()
        {
            Initialize();
        }

        private void Initialize()
        {
            this.root = null;
        }

        public void Add(int e)
        {
            root = Add(root, e);
        }

       public int Question2(BNode n)
       {
               // write your implementation here
       }
}
```

3. You have to implement Stack ADT. The stack ADT must have following functions:

STACK ADT

Push()

Pop()

IsEmpty()

Our implementation shouldn't restrict us the number of elements that can be inserted.

```
class StackADT
{
        // your implementation
}
```