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 and then print the numbers in descending order. You can use any sorting algorithm but your code should run in O(log n). The fastest algorithm will be given the highest marks. The pseudo-code of common sorting algorithms are given below.

Following is the sample output for n = 4

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

7
2
1
-5
```

Following is the sample output for n = 3

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

17
12
1
```

Pseudo-codes:

```
Selection Sort:
     For I = 0 to N-1 do:
       Smallsub = I
       For J = I + 1 to N-1 do:
         If A(J) < A(Smallsub)
           Smallsub = J
         End-If
       End-For
       Temp = A(I)
       A(I) = A(Smallsub)
       A(Smallsub) = Temp
     End-For
```

```
Insertion Sort:
     For I = 1 to N-1
       J = I
       Do while (J > 0) and (A(J) < A(J - 1)
         Temp = A(J)
         A(J) = A(J - 1)
```

```
          A(J - 1) = Temp
          J = J - 1
        End-Do
      End-For
```

```
Bubble Sort:
      For I = 0 to N - 2
        For J = 0 to N - 2
          If (A(J) > A(J + 1)
            Temp = A(J)
            A(J) = A(J + 1)
            A(J + 1) = Temp
          End-If
        End-For
      End-For
```

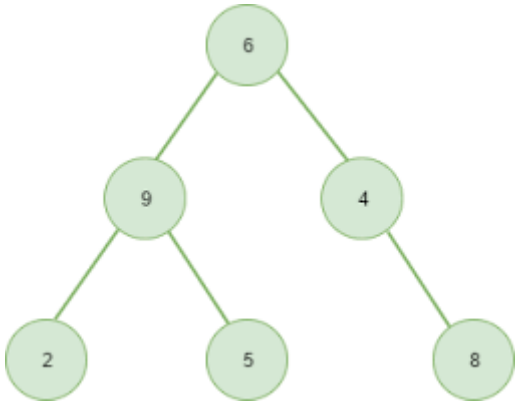
```
QUICKSORT(A, p, r)
1  if p < r
2          q ← PARTITION(A, p, r)
3          QUICKSORT(A, p, q − 1)
4          QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
1  i ← p − 1
2  for j ← p to r − 1
3          if A[j] ≤ A[r]
4                  i ← i + 1
5                  swap A[i] and A[j]
6  swap A[i + 1] and A[r]
7  return i + 1
```

2. You have to write a function to size of the given node. The size of a node is: the number of descendants the node has (including the node itself). The size of root is the size of a tree. The size of a leaf is 1.the length of the path form the node to the deepest leaf. The pseudo-code of the recursive function of height is given bellow:

```
Size: Given n = node
        If n is not an empty tree
                Size = 1 +Size (n -> left) + Size(n-> right)
```

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



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 Queue ADT. The Queue ADT must have following functions:

STACK ADT
Enqueue() Dequeue() Count()

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

```
class QueueADT
{
        // your implementation
}
```