Status	Completed
Attempt Score	20 out of 20 points

QUESTION 6

In a binary search tree, the attempt of inserting a key which already exists in the tree will result in:

- (•) Insertion will fail since we can't have two equal keys in the same tree
- ( ) A new node will be created and inserted right of that node with equal key
- ( ) The node with equal key is updated with new data
- ( ) A new node will be created and inserted left of that node with equal key
- ( ) None

QUESTION 5

In a binary search tree, if a key is deleted and it has two sub-trees, then:

1. It can be replaced with the min-key in the right sub-tree, before deleting that min-key
2. It can be replaced with the max-key in the left sub-tree, before deleting that max-key
3. It can be only replaced with the min-key in the right sub-tree, before deleting that min-key
4. Both 1 and 2 are correct
5. None

QUESTION 7

In a binary search tree, the traversal method which can be used to print the keys in increasing order is:

- Pre-order
- In-order
- Infix
- None
- Post-order

QUESTION 3

A subtree is:

- A child node in a tree
- A tree that starts from the root of a tree to a certain level in the tree
- A tree that consist of a node N and its descendants. Node N can't be the root node
- A tree that consist of a node N and its descendants. Node N can be the root node
- None

QUESTION 18

Recursion is:

- A class
- a process of defining a method that calls other methods repeatedly
- a process of defining a method that calls itself repeatedly
- a process of defining a method that calls other methods which in turn call again this method

QUESTION 19

Which of these will happen if a recursive method does not have a base case?

- A compile-time error
- The system stops the program after 100 of calls
- An infinite recursion until an exception occurs
- None

QUESTION 8

In a binary search tree, the average-case time-complexity in general for the search and insert operations:

- (•) Both O(logn)
- ( ) Both O(n)
- ( ) O(logn) and O(n) respectively.
- ( ) O(logn) and O(nlogn), respectively

QUESTION 9

Suppose the keys 7, 5, 1, 8, 3 are inserted in that order into an initially empty binary search tree. The pre-order traversal sequence of the resulted tree is:

- None
- 7, 5, 3, 1, 8
- 3, 1, 5, 8, 7
- 7, 5, 1, 3, 8
- 1, 3, 5, 7, 8

QUESTION 4

What is the resulting binary tree after the following code.

```
BT<Integer> bt = new BT<Integer>();
bt.insert(Relative.Root, 10);
bt.insert(Relative.LeftChild, 15);
bt.insert(Relative.RightChild, 30);
```

- 10 will be root; 15 will be 10's left child and 30 will be 10's right child
- 15 can't be inserted left of 10 because 15 is larger than 10
- 10 will be root; 30 will be 10's right child and 15 will be 30's left child
- None
- 10 will be root; 15 will be 10's left child and 30 will be 15's right child

QUESTION 2

What does this method do?

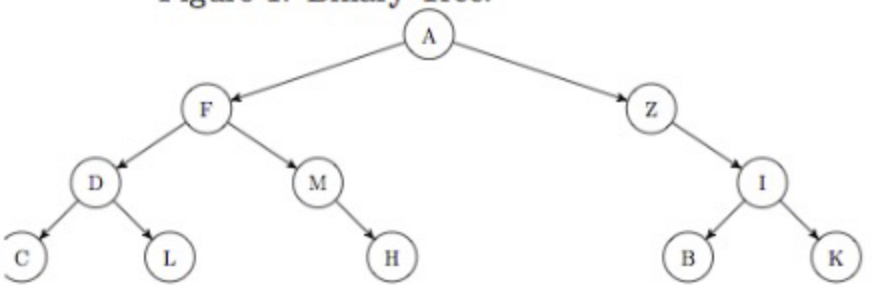
```
private BTNode method(T p){
  LinkStack<BTNode > stack = new LinkStack<BTNode <T>>();
  BTNode<T> q = root;
  while(q != null && q.data != p){
        if(q.left != null)
              stack.push(q.left);
        if(q.right != null)
              q = q.right;
        else
              if(!stack.empty())
                q = stack.pop();
              else
                 q=null;

   }
  return q;
}
```

- Returns null
- The program will crash because of the while loop's condition
- Returns p's parent node
- Looks for p and returns the node that has it
- Will go all the way to the right of the tree, then return null

QUESTION 17

Indicate the **postorder** traversals of the tree 1. shown in Figure 1

Figure 1: Binary Tree.



- CLDHMFBKIZA
- CBLHKDMIFZA
- ACDLFMHBIKZ
- CDLFMHAZBIK

QUESTION 16

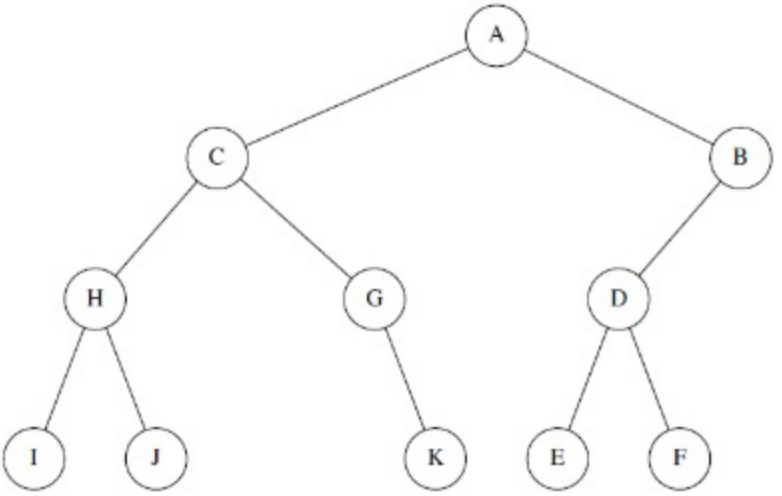


Figure 1: Binary tree

what is the result of running func() on binary tree in Figure 1

```
public int func () {   return rec_func(root);  }
private int rec_func (BTNode<T>  t)
{
if (t == null)
 return 0;
if (t.right != null && t.left != null)
   return 1+rec_func(t.right)+rec_func(t.left);
else
    return rec_func(t.right)+rec_func(t.left);    }
```

- 6
- 11
- 4
- 5

QUESTION 1

For a method in BinaryTree ADT, fill in the blanks for the method print and print_rec. The method will print the element of the tree using "in-order" order.

```
public void print(){
 printRec( root );
}
private void printRec( BTNode<T> p){
 if ( p == null )
 return;
 printRec( p.left );
 System.out.print( p.data );
 printRec( p.right );
}
```

QUESTION 10

Consider a member method that prints the keys of a BST in a reverse order, complete the following code by choosing the correct answer:

```
public void printReverse() {
    1. ...
}
private void recPrintReverse(BSTNode<T> n) {
    2. ...
            3. ...
    4. ...
    5. ...
    6. ...
}
```

Line 1:

- return recPrintReverse(root);
- None
- recPrintReverse(root.right);
- recPrintReverse(root.left);
- recPrintReverse(root);

QUESTION 11

```
public void printReverse() {

        1. ...

   }

   private void recPrintReverse(BSTNode<T> n) {

        2. ...

                3. ...

        4. ...

        5. ...

        6. ...

   }
```

Line 2:

- if(n != null)
- None
- if(root != null)
- if(n == null)
- if(root == null)

QUESTION 12

```
public void printReverse() {

        1. ...

   }

   private void recPrintReverse(BSTNode<T> n) {

        2. ...

                3. ...

        4. ...

        5. ...

        6. ...

   }
```

Line 3:

- return;
- None
- return n;
- n = n.left;
- n = n.right;

QUESTION 13

```
public void printReverse() {
        1. ...
   }
   private void recPrintReverse(BSTNode<T> n) {
        2. ...
                3. ...
        4. ...
        5. ...
        6. ...
   }
```

Line 4:

- recPrintReverse(n.right);
- recPrintReverse(root);
- recPrintReverse(n.left);
- None
- recPrintReverse(n)

QUESTION 14

```
public void printReverse() {
        1. ...
    }
    private void recPrintReverse(BSTNode n){
        2. ...
                3. ...
        4. ...
        5. ...
        6. ...
    }
```

Line 5:

- System.out.println(n.left.key);
- None
- System.out.println(n.key);
- System.out.println(root.key);
- System.out.println(n.right.key);

QUESTION 15

```
public void printReverse() {

        1. ...

    }

    private void recPrintReverse(BSTNode  n){

        2. ...

                3. ...

        4. ...

        5. ...

        6. ...

    }
```

Line 6:

- (●) recPrintReverse(n.left);
- ( ) recPrintReverse(root);
- ( ) None
- ( ) recPrintReverse(n.right);
- ( ) recPrintReverse(n);