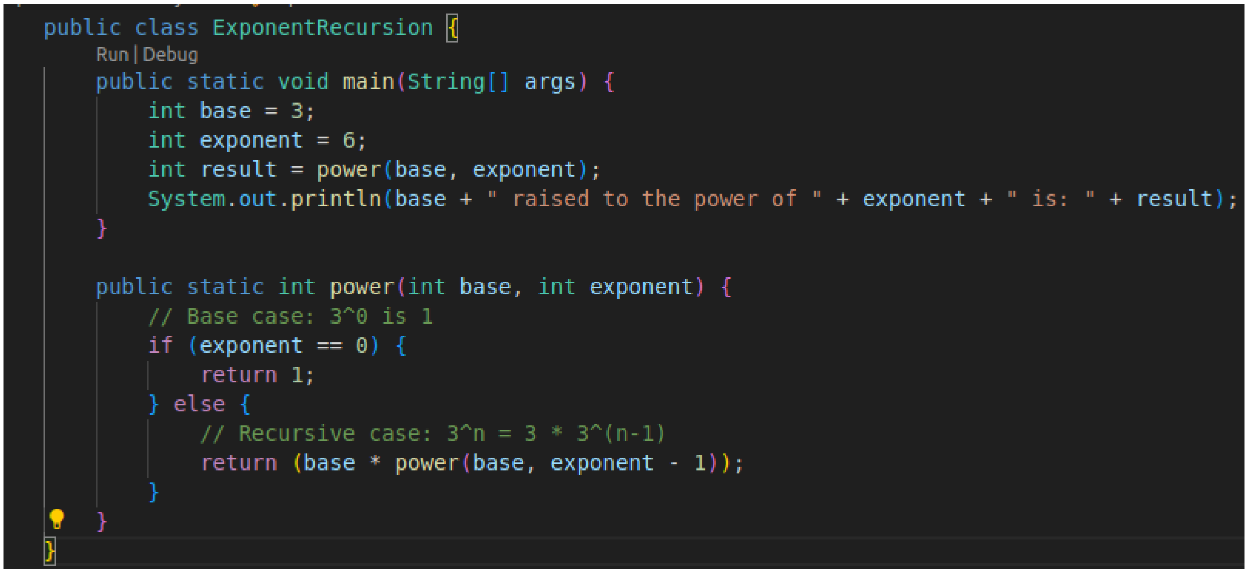
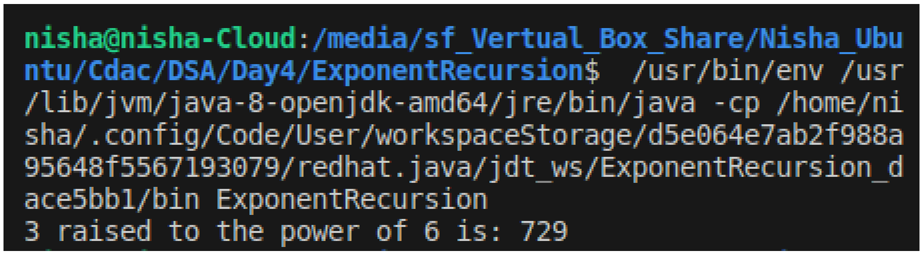
Lab Assignment [DSA] - 4

Q1.Write a java program to find the value of 3 raised to 6 using recursion.



```java
public class ExponentRecursion {
    public static void main(String[] args) {
        int base = 3;
        int exponent = 6;
        int result = power(base, exponent);
        System.out.println(base + " raised to the power of " + exponent + " is: " + result);
    }

    public static int power(int base, int exponent) {
        // Base case: 3^0 is 1
        if (exponent == 0) {
            return 1;
        } else {
            // Recursive case: 3^n = 3 * 3^(n-1)
            return (base * power(base, exponent - 1));
        }
    }
}
```



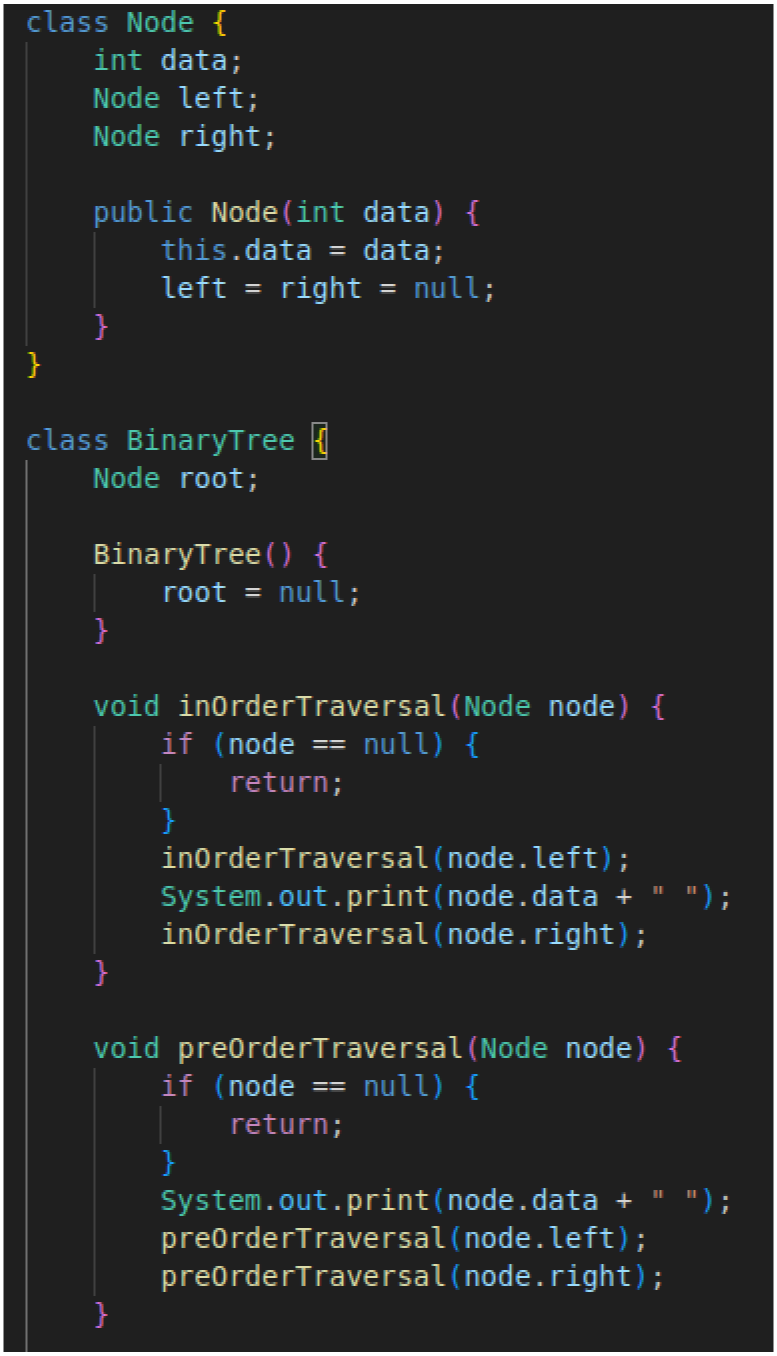
```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubu
ntu/Cdac/DSA/Day4/ExponentRecursion$  /usr/bin/env /usr
/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/ni
sha/.config/Code/User/workspaceStorage/d5e064e7ab2f988a
95648f5567193079/redhat.java/jdt_ws/ExponentRecursion_d
ace5bb1/bin ExponentRecursion
3 raised to the power of 6 is: 729
```

Q2. Using above Fig, find the Traversal order for each case

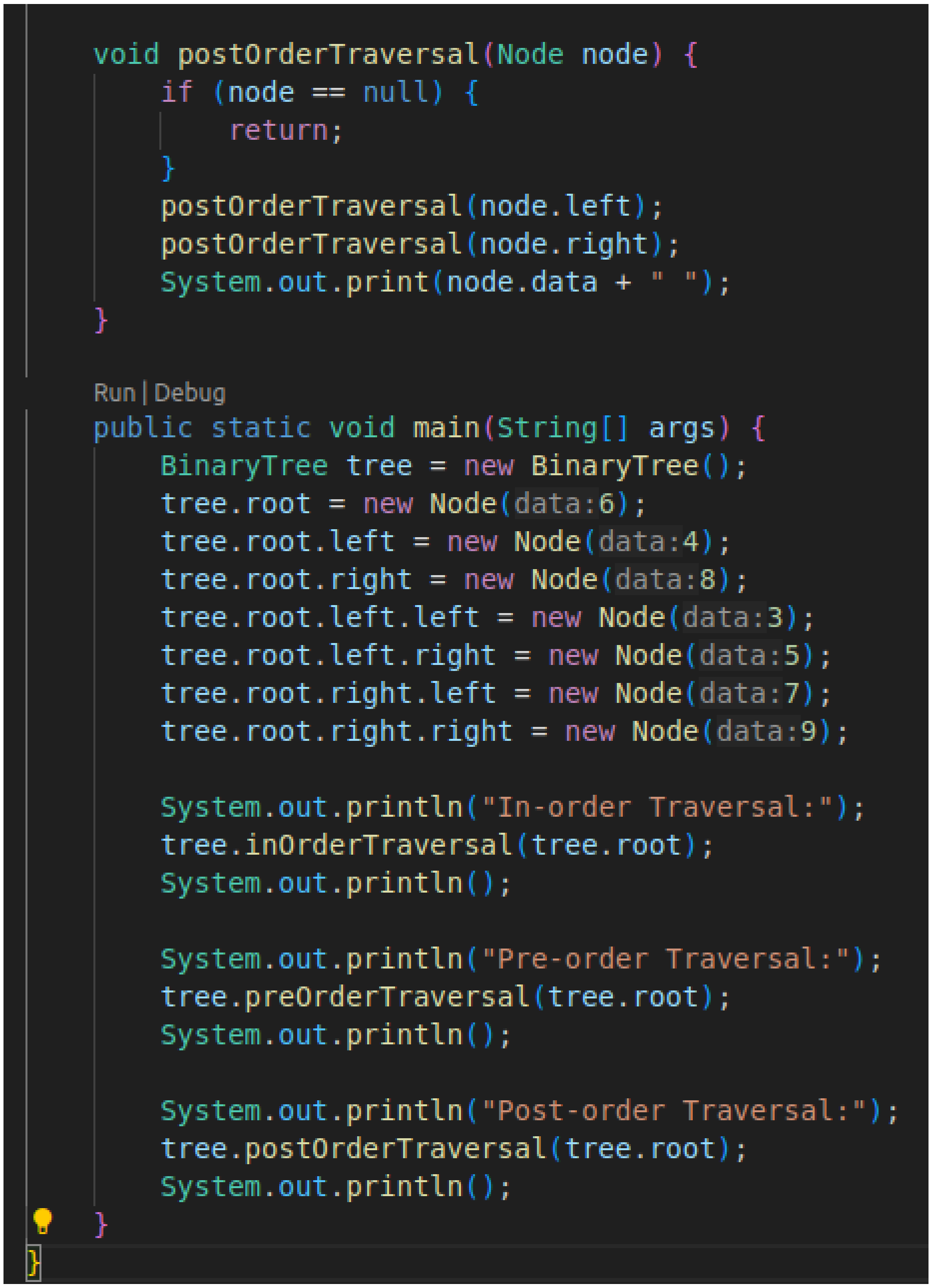
1 In-order Traversal

2 In pre-order Traversal

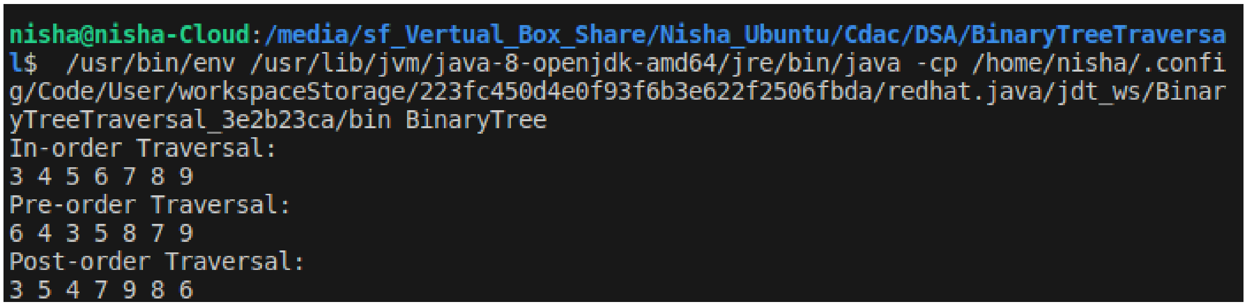
3 In Post-order



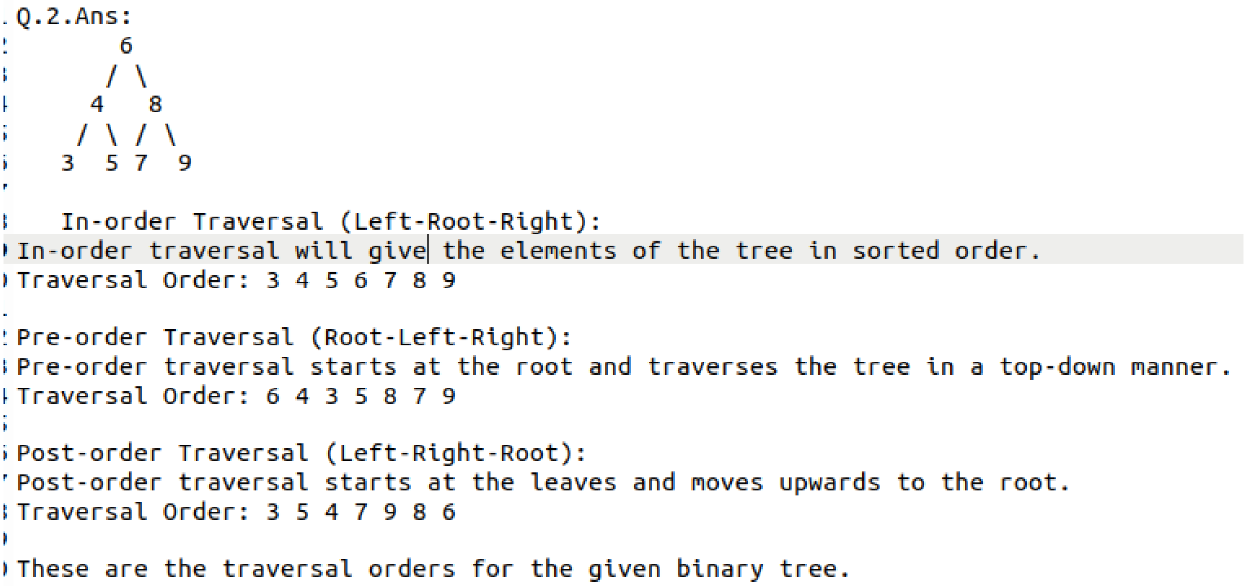
```java
class Node {
    int data;
    Node left;
    Node right;

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

class BinaryTree {
    Node root;

    BinaryTree() {
        root = null;
    }

    void inOrderTraversal(Node node) {
        if (node == null) {
            return;
        }
        inOrderTraversal(node.left);
        System.out.print(node.data + " ");
        inOrderTraversal(node.right);
    }

    void preOrderTraversal(Node node) {
        if (node == null) {
            return;
        }
        System.out.print(node.data + " ");
        preOrderTraversal(node.left);
        preOrderTraversal(node.right);
    }
```



```
    void postOrderTraversal(Node node) {
        if (node == null) {
            return;
        }
        postOrderTraversal(node.left);
        postOrderTraversal(node.right);
        System.out.print(node.data + " ");
    }

    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(data:6);
        tree.root.left = new Node(data:4);
        tree.root.right = new Node(data:8);
        tree.root.left.left = new Node(data:3);
        tree.root.left.right = new Node(data:5);
        tree.root.right.left = new Node(data:7);
        tree.root.right.right = new Node(data:9);

        System.out.println("In-order Traversal:");
        tree.inOrderTraversal(tree.root);
        System.out.println();

        System.out.println("Pre-order Traversal:");
        tree.preOrderTraversal(tree.root);
        System.out.println();

        System.out.println("Post-order Traversal:");
        tree.postOrderTraversal(tree.root);
        System.out.println();
    }
}
```



```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/BinaryTreeTraversa
l$  /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.confi
g/Code/User/workspaceStorage/223fc450d4e0f93f6b3e622f2506fbda/redhat.java/jdt_ws/Binar
yTreeTraversal_3e2b23ca/bin BinaryTree
In-order Traversal:
3 4 5 6 7 8 9
Pre-order Traversal:
6 4 3 5 8 7 9
Post-order Traversal:
3 5 4 7 9 8 6
```

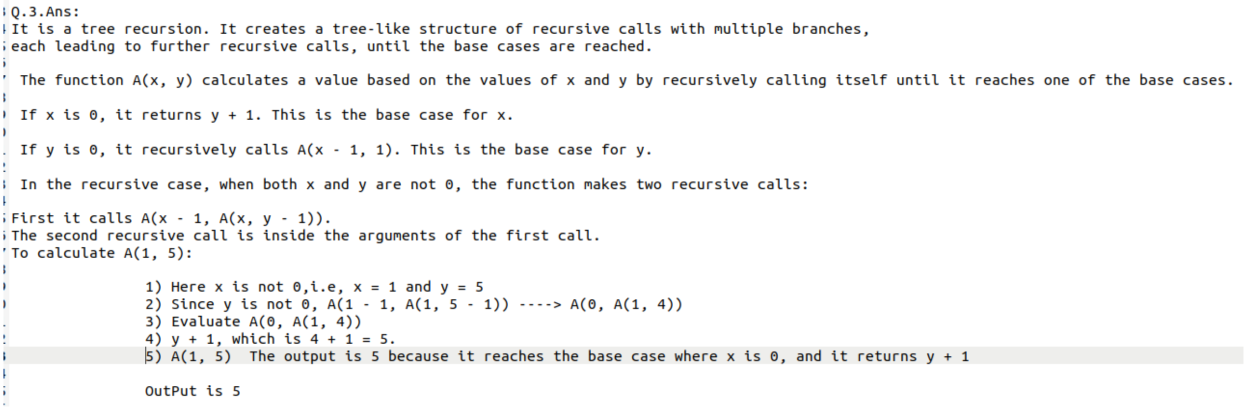


```
Q.2.Ans:
       6
      / \
     4   8
    / \ / \
   3  5 7  9

   In-order Traversal (Left-Root-Right):
In-order traversal will give the elements of the tree in sorted order.
Traversal Order: 3 4 5 6 7 8 9

Pre-order Traversal (Root-Left-Right):
Pre-order traversal starts at the root and traverses the tree in a top-down manner.
Traversal Order: 6 4 3 5 8 7 9

Post-order Traversal (Left-Right-Root):
Post-order traversal starts at the leaves and moves upwards to the root.
Traversal Order: 3 5 4 7 9 8 6

These are the traversal orders for the given binary tree.
```

Q3. Consider the following recursion function.

```
A(x, y)
{
if(x==0)
{
return (y+1);
}
if(y==0)
{
return (A(x-1,1));
}
else
{
return (A(x-1,A(x,y-1)));
}
}
```

What is the output______. if A(1,5) is called? Explain the Concept.



Q.3.Ans:
It is a tree recursion. It creates a tree-like structure of recursive calls with multiple branches,
each leading to further recursive calls, until the base cases are reached.

The function A(x, y) calculates a value based on the values of x and y by recursively calling itself until it reaches one of the base cases.

If x is 0, it returns y + 1. This is the base case for x.

If y is 0, it recursively calls A(x - 1, 1). This is the base case for y.

In the recursive case, when both x and y are not 0, the function makes two recursive calls:

First it calls A(x - 1, A(x, y - 1)).
The second recursive call is inside the arguments of the first call.
To calculate A(1, 5):

```
        1) Here x is not 0,i.e, x = 1 and y = 5
        2) Since y is not 0, A(1 - 1, A(1, 5 - 1)) ----> A(0, A(1, 4))
        3) Evaluate A(0, A(1, 4))
        4) y + 1, which is 4 + 1 = 5.
        5) A(1, 5)  The output is 5 because it reaches the base case where x is 0, and it returns y + 1

        OutPut is 5
```