DSA LAB EXAM

1. Write a Java program to

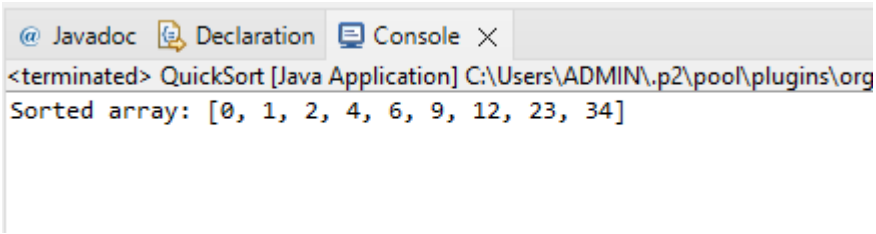
a. Perform quick sort

Solution :

```
package com.labexam;

import java.util.Arrays;

public class QuickSort {

public static void main(String[] args) {

int[] array = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };

quickSort(array, 0, array.length - 1);

System.out.println("Sorted array: " + Arrays.toString(array));

}

public static void quickSort(int[] array, int low, int high) {

if (low < high) {

int pivotIndex = partition(array, low, high);

quickSort(array, low, pivotIndex);

quickSort(array, pivotIndex + 1, high);

}

}

public static int partition(int[] array, int low, int high) {

int pivot = array[low];

int i = low - 1;

int j = high + 1;

while (true) {

do {

i++;

} while (array[i] < pivot);
```

```
do {
j--;
} while (array[j] > pivot);
if (i >= j) {
return j;
}
int temp = array[i];
array[i] = array[j];
array[j] = temp;
}
}
}
```

Output :



b. Perform preorder tree traversal

Solution :

```
package com.labexam;
import java.util.Stack;
class Node {
int data;
Node left;
Node right;
public Node(int data) {
```

```
this.data = data;
left = null;
right = null;
}
}
public class PreorderTraversal {
public static void preOrderTraversal(Node root) {
if (root == null) {
return;
}
System.out.print(root.data + " ");
preOrderTraversal(root.left);
preOrderTraversal(root.right);
}
public static void preOrderTraversalIterative(Node root) {
if (root == null) {
return;
}
Stack<Node> stack = new Stack<>();
stack.push(root);
while (!stack.isEmpty()) {
Node current = stack.pop();
System.out.print(current.data + " ");
if (current.right != null) {
stack.push(current.right);
}
if (current.left != null) {
```

```
stack.push(current.left);

}

}

}

public static void main(String[] args) {

Node root = new Node(1);

root.left = new Node(2);

root.right = new Node(3);

root.left.left = new Node(4);

root.left.right = new Node(5);

System.out.print("Recursive Preorder traversal : ");

preOrderTraversal(root);

System.out.println();

System.out.print("Iterative Preorder traversal : ");

preOrderTraversalIterative(root);

}

}
```

Output :