Name : Madhavi Suratkar

PNR : 220960920081

LAB EXAM

DATA STRUCTURE AND ALGORITHMS

1. Write a Java program to

a. Perform binary search operation

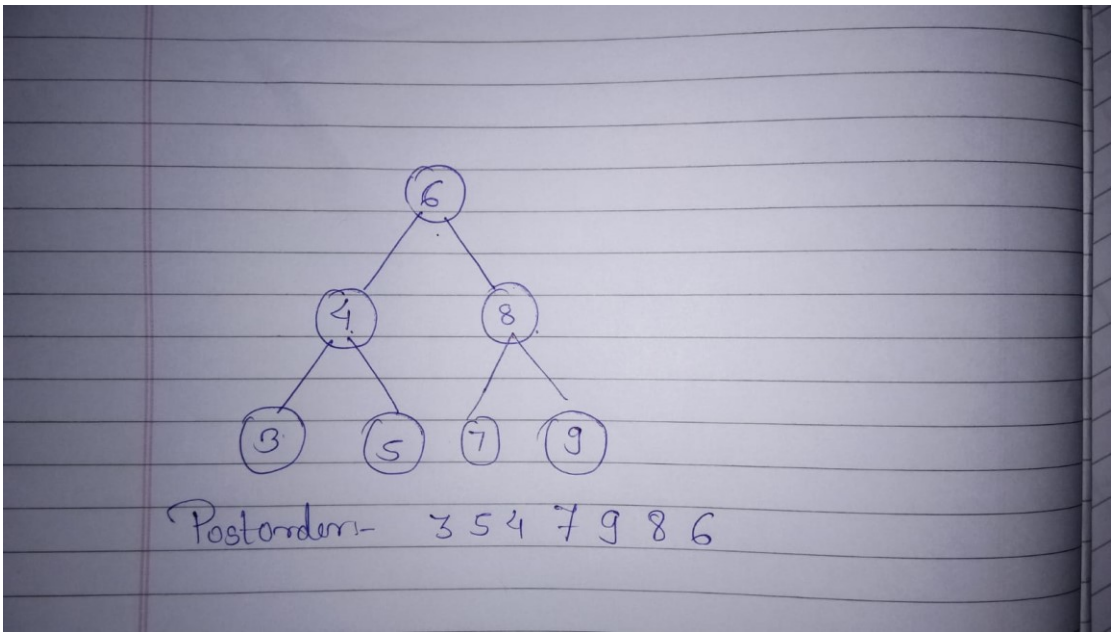
```
package com.entiry;


public class BinarySearchOperation
{


        public static void binarySearch(int arr[], int first, int last, int key){
          int mid = (first + last)/2;
          while( first <= last ){
            if ( arr[mid] < key ){
              first = mid + 1;
            }else if ( arr[mid] == key ){
              System.out.println("Element is found at index: " + mid);
              break;
            }else{
               last = mid - 1;
            }
            mid = (first + last)/2;
          }
          if ( first > last ){
            System.out.println("Element is not found!");
          }
        }
        public static void main(String args[]){
              int arr[] = {10,20,30,40,50};
              int key = 30;
              int last=arr.length-1;
              binarySearch(arr,0,last,key);
        }
       }
```

```
Console ×  web.xml  LifeCycleDemo/pom.xml  Array_test.java  AppMain.java  BinarySearch.java  BinarySearchOperation.java
<terminated> BinarySearchOperation [Java Application] D:\TVM_CDAC\JAVA\Java_Program\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220
Element found at index : 2
```

b. Execute tree traversal in postorder

```
package com.entiry;

import java.util.Scanner;

public  class BinarySearch
{

	static class Node{
		int data;
		Node left=null;
		Node right=null;

		public Node(int data) {
			this.data=data;
		}
	}
	static Node createTree() {
		int data;
		Node root=null;

	  Scanner sc=new Scanner(System.in);

	  System.out.print("Enter data : ");
	  data=sc.nextInt();

	  if(data == -1)
	  {
		return null;
	  }
	  root=new Node(data);

	  System.out.println("Enter left child of the node "+root.data);
	  root.left=createTree();

	  System.out.println("Enter right child of the node "+root.data);
	  root.right=createTree();
//	  sc.close();
	  return root;
	}




	static void postorder(Node root) {
		if(root == null)
```

```
                return;
        postorder(root.left);
                postorder(root.right);
                System.out.print(root.data +" ");
        }



                public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        Node root=createTree();

        System.out.println();

        System.out.println("Postorder");
        postorder(root);

        System.out.println();



    }
}
```



```
<terminated> BinarySearch [Java Application] D:\TVM_CDAC\JAVA\Java_Program\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\ja
Enter data : 6
Enter left child of the node 6
Enter data : 4
Enter left child of the node 4
Enter data : 3
Enter left child of the node 3
Enter data : -1
Enter right child of the node 3
Enter data : -1
Enter right child of the node 4
Enter data : 5
Enter left child of the node 5
Enter data : -1
Enter right child of the node 5
Enter data : -1
Enter right child of the node 6
Enter data : 8
Enter left child of the node 8
Enter data : 7
Enter left child of the node 7
Enter data : -1
Enter right child of the node 7
Enter data : -1
Enter right child of the node 8
Enter data : 9
Enter left child of the node 9
Enter data : -1
Enter right child of the node 9
Enter data : -1
```

```
<terminated> BinarySearch [Java Application] D:\TVM_CDAC\JAVA\Java_Program\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\b
Enter left child of the node 3
Enter data : -1
Enter right child of the node 3
Enter data : -1
Enter right child of the node 4
Enter data : 5
Enter left child of the node 5
Enter data : -1
Enter right child of the node 5
Enter data : -1
Enter right child of the node 6
Enter data : 8
Enter left child of the node 8
Enter data : 7
Enter left child of the node 7
Enter data : -1
Enter right child of the node 7
Enter data : -1
Enter right child of the node 8
Enter data : 9
Enter left child of the node 9
Enter data : -1
Enter right child of the node 9
Enter data : -1

Postorder
3 5 4 7 9 8 6
```