LAB EXAM

DATA STRUCTURE AND ALGORITHMS

Total Marks: 20 Time: 90 MIN

PRN: 220960920086
Name: Sudarshan Rajendra Ukharde

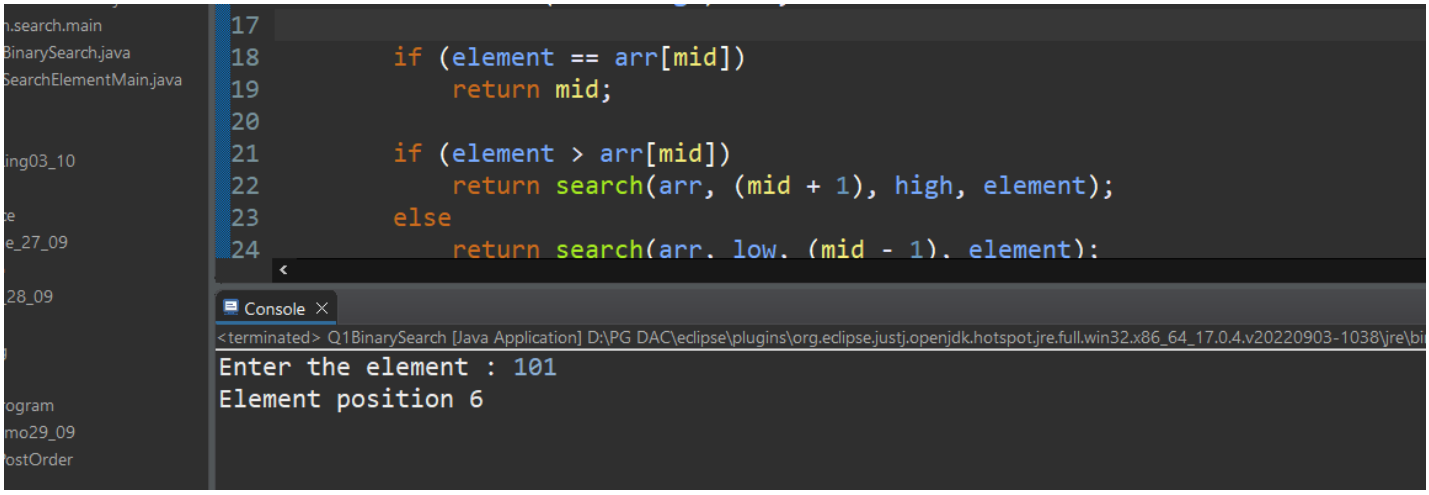
1.Write a Java program to

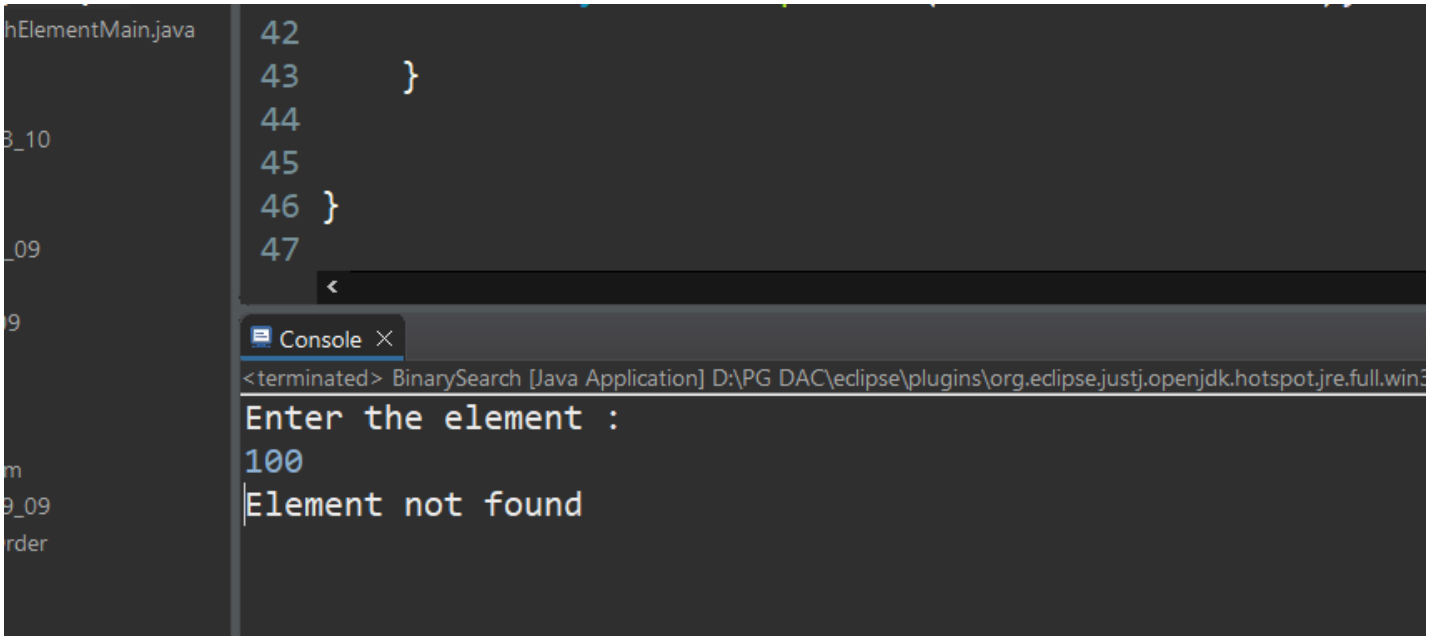
1. **Perform binary search operation**

```
package com.binarysearch;

import java.util.Scanner;

public class Q1BinarySearch {

	public int search(int arr[], int low, int high, int element)
	{
		if(low == high) {
			if(arr[low] == element)
				return low;
			else
				return -1;
		}

		int mid = (low + high) / 2;

		if (element == arr[mid])
			return mid;

		if (element > arr[mid])
			return search(arr, (mid + 1), high, element);
		else
			return search(arr, low, (mid - 1), element);
	}


	public static void main(String[] args) {

		Q1BinarySearch obj = new Q1BinarySearch();
		int arr[] = {89, 85, 96, 52, 69, 101};
```

```
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter the element : ");
            int element = sc.nextInt();

            int n = arr.length - 1;

            int position = obj.search(arr, 0, n, element);
            if(position >=0)
                  System.out.println("Element position "+(position+1));
            else
                  System.out.println("Element not found");

      }

}
```

Output:





2. **Execute tree traversal in postorder.**

Node.java

```
package com.postordertraversal;

public class Node {
      int data;
    Node left, right;

    Node(int item) {
        data = item;
        left = right = null;
    }

}
```

Binarytree.java

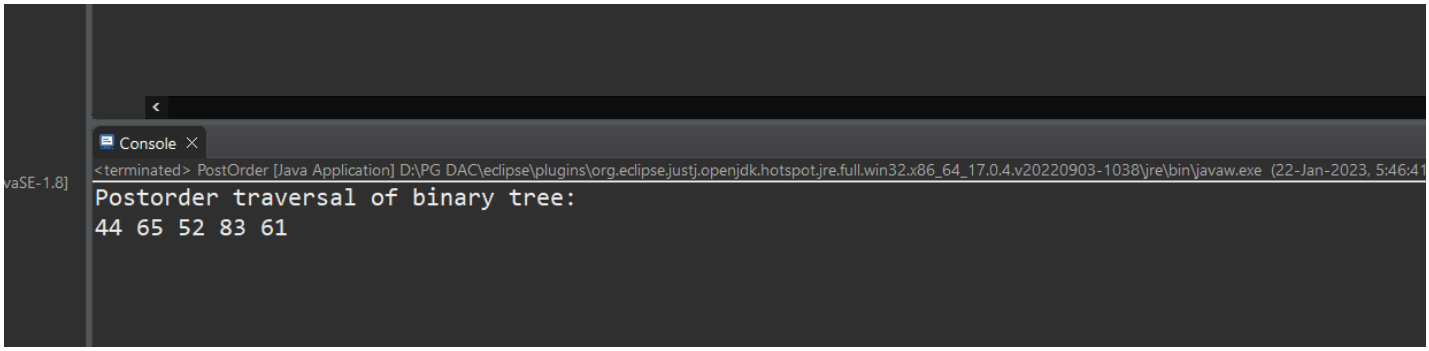
```
package com.postordertraversal;

public class BinaryTree {
       Node root;

          void postorder() {
              postorder(root);
          }

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

PostOrder.java

```
package com.postordertraversal;

public class PostOrder {

      public static void main(String[] args) {
            BinaryTree tree = new BinaryTree();
        tree.root = new Node(61);
        tree.root.left = new Node(52);
        tree.root.right = new Node(83);
        tree.root.left.left = new Node(44);
        tree.root.left.right = new Node(65);
        System.out.println("Postorder traversal of binary tree:");
```

```
        tree.postorder();
    }

}
```

Output:



* * * * * * * * * *