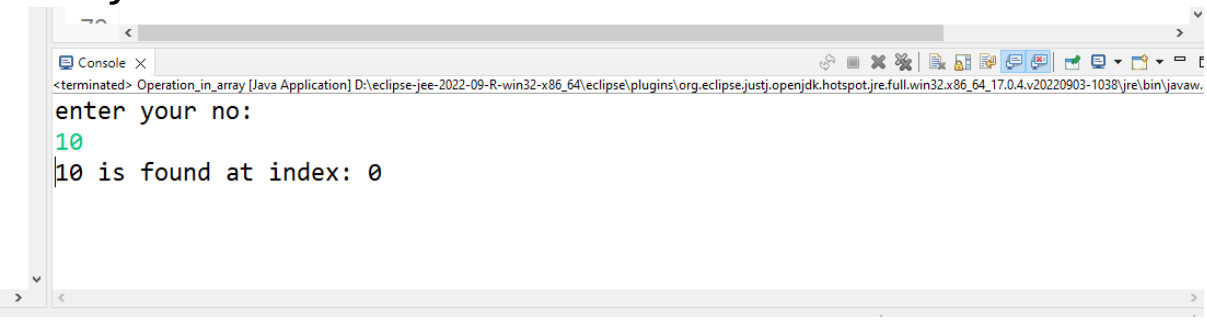
1. Write a Java program to

a. Search an item through linear search

```
import java.util.Scanner;

public class Operation_in_array {

public static int linearSearch(int[] a1, int
num){

     for(int i=0;i<a1.length;i++){

             if(a1[i] == num){
              return i;
               }

            }

           return -1;
         }
                public static void main(String
a[]){
                    int[] a1=
{10,20,30,50,70,90};
                  Scanner sc= new
Scanner(System.in);
                  System.out.println("enter
your no:");
                  int num = sc.nextInt();



                   System.out.println (num+ "
is found at index: " +linearSearch(a1, num));
```

```
        }

    }
```



2

```
import java.io.*;

class BinarySearchTree {

    class Node {
        int key;
        Node left, right;

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


    Node root;
```

```
            BinarySearchTree() { root = null;
}


            void deleteKey(int key) { root =
deleteRec(root, key); }


            Node deleteRec(Node root, int key)
            {
            if (root == null)
                    return root;
                if (key < root.key)
                    root.left =
deleteRec(root.left, key);
                else if (key > root.key)
                    root.right =
deleteRec(root.right, key);



                else {

                    if (root.left == null)
                        return root.right;
                    else if (root.right ==
null)
                        return root.left;


                    root.key =
minValue(root.right);
```

```
                    root.right =
deleteRec(root.right, root.key);
                }

                return root;
            }

            int minValue(Node root)
            {
                int minv = root.key;
                while (root.left != null) {
                    minv = root.left.key;
                    root = root.left;
                }
                return minv;
            }


            void insert(int key) { root =
insertRec(root, key); }


            Node insertRec(Node root, int key)
            {


                if (root == null) {
                    root = new Node(key);
                    return root;
                }
```

```
                if (key < root.key)
                    root.left =
insertRec(root.left, key);
                else if (key > root.key)
                    root.right =
insertRec(root.right, key);

                return root;
            }


            void inorder() { inorderRec(root);
}


            void inorderRec(Node root)
            {
                if (root != null) {
                    inorderRec(root.left);
                    System.out.print(root.key +
" ");
                    inorderRec(root.right);
                }
            }


            public static void main(String[]
args)
            {
                BinarySearchTree tree = new
BinarySearchTree();

                    tree.insert(50);
```

```
                    tree.insert(30);
                    tree.insert(20);
                    tree.insert(40);
                    tree.insert(70);
                    tree.insert(60);
                    tree.insert(80);

                System.out.println(
                    "Inorder traversal of the
given tree");
                tree.inorder();

                System.out.println("\nDelete
20");
                tree.deleteKey(20);



            }



    }
```

output

```
Inorder traversal of the given tree
20304050607080
Delete 20
```