Lab3

```java
/**
 * Created by lenovo on 23/01/2022.
 */
public class ArrayStack<E> implements Stack<E>{
    E data[];
    int t=-1;
    static final int CAPACITY=100;

    public ArrayStack(int c){
        data=(E[])new Object[c];
    }

    public ArrayStack() {
        this(CAPACITY);
    }

    @Override
    public boolean isEmpty() {
        return t==-1;
    }

    @Override
    public int size() {
        return t+1;
    }

    @Override
    public void push(E element) {
        if(size()==data.length)
            throw new IllegalStateException("Stack is full");
           // t++;
            data[++t]=element;

    }

    @Override
    public E pop() {
        if(isEmpty())
        return null;

        E x=data[t];
        data[t]=null;
        t--;
        return x;
    }

    @Override
    public E top() {
        if(isEmpty())
        return null;
        return data[t];
    }
}
```

```java
import java.util.Arrays;

/**
 * Created by lenovo on 23/01/2022.
 */
public class ReversUsingStack {
    public static void main(String[] args) {
        int []a={11,12,13,14,15};
        System.out.println(Arrays.toString(a));
        ArrayStack<Integer>s=new ArrayStack<>(a.length);

        for (int i = 0; i < a.length; i++) {
            s.push(a[i]);

        }
        for (int i = 0; i <a.length ; i++) {
            a[i]=s.pop();
        }
        System.out.println("after revers using stack");
        System.out.println(Arrays.toString(a));
    }
}
```

```java
/**
 * Created by lenovo on 23/01/2022.
 */
public interface Stack<E> {
    boolean isEmpty();
    int size();
    void push(E element);
    E pop();
    E top();


}
```

```java
import java.util.Scanner;

/**
 * Created by lenovo on 23/01/2022.
 */
public class TestStack {
    public static void main(String[] args) {
        ArrayStack<Integer>myStack=new ArrayStack<>(5);
        Scanner in=new Scanner(System.in);

        System.out.println("is stack empty? "+myStack.isEmpty());
        System.out.println("input element");

        for (int i = 0; i < 5; i++) {
            myStack.push(in.nextInt());

            System.out.println("top element is= "+myStack.top());
            System.out.println("size of stack is "+myStack.size());
            System.out.println("is the stack empty? "+myStack.isEmpty());


        }

        for (int i = 0; i < 6; i++) {
            System.out.println("deleted element is "+myStack.pop());
            System.out.println("top element after delete= "+myStack.top());
            System.out.println("size of stack is "+myStack.size());
            System.out.println("is the stack empty? "+myStack.isEmpty());


        }
    }
}
```

R-6.4 Implement a method with signature transfer(S, T) that transfers all elements from stack S onto stack T, so that the element that starts at the top of S is the first to be inserted onto T, and the element at the bottom of S ends up at the top of T

```
import java.util.Stack;

/**
 * Created by USER on 31/01/2022.
 */
public class Transfer {
    public static Stack<Integer>stackpush(Stack<Integer>stack){
        for (int i=0;i<6;i++){
            Integer push=(Integer)stack.push(i);
            System.out.println(push);
        }
        return stack;

    }
    public static void pop(Stack<Integer>stack){
        for (int i=0;i<6;i++){
            Integer pop=(Integer)stack.pop();
            System.out.println(pop);
        }
    }
    public static void peek(Stack<Integer> stack){

        Integer peek = (Integer) stack.peek();

        System.out.println("Top of the element is: " + peek);

    }


    public static void search(Stack<Integer> stack, int element){

        System.out.println("Element searched is: ");

        Integer search = (Integer) stack.search(element);
```

```java
        if(element == -1){

            System.out.println("Stack is empty.");
        }
        else{
            System.out.println("Element is: " + search);
        }


    }

    public static Stack<Integer> transfer(Stack<Integer> stack1, Stack<Integer>
stack2){


        stack2 = stackPush(stack1);



        System.out.println("Stack transfered successfully: "+ stack2);

        return stack2;

    }

    protected static Stack<Integer> stackPush(Stack<Integer> stack1) {
    }

        public static void main (String[]args){

            Stack<Integer> stack1 = new Stack<>();
            Stack<Integer> stack2 = new Stack<>();

            Stack<Integer> finalStack = transfer(stack1, stack2);


        }




    }
```