Data Structure Lab7 : Stack 2022-2023

1. **Create Stack Interface**
   Define a `Stack` interface with common stack methods:

   ```
   java
   Copy code
   public interface Stack<E> {
       boolean isEmpty();
       int size();
       E top();
       void push(E e);
       E pop();
   }
   ```

2. **Create Stack Using Array**
   Implement the `Stack` interface using an array.
3. **Create Stack Using Linked Lists**
   Implement the `Stack` interface using a linked list.
4. **Implement Basic Methods of Stack**
   Ensure the methods `isEmpty()`, `size()`, `top()`, `push(E e)`, and `pop()` work properly.

Homework

1. Implement the `transfer(S, T)` Method

Transfer all elements from stack `S` to stack `T` such that the top of `S` ends up at the top of `T`.

```
java
Copy code
public class StackTransfer {
    public static <E> void transfer(Stack<E> S, Stack<E> T) {
        while (!S.isEmpty()) {
            T.push(S.pop());
        }
    }
}
```

Data Structure Lab7 : Stack 2022-2023

2. Recursive Method to Remove All Elements from a Stack

Define a recursive function to clear a stack.

```
java
Copy code
public class RecursiveClearStack {
    public static <E> void clear(Stack<E> stack) {
        if (!stack.isEmpty()) {
            stack.pop();
            clear(stack);
        }
    }
}
```

3. Evaluate an Expression in Postfix Notation (Non-Recursive)

A stack-based approach to evaluate postfix expressions.

```
java
Copy code
import java.util.Stack;

public class PostfixEvaluator {
    public static int evaluate(String expression) {
        Stack<Integer> stack = new Stack<>();
        String[] tokens = expression.split(" ");

        for (String token : tokens) {
            if (token.matches("-?\\d+")) { // Check if token is a number
                stack.push(Integer.parseInt(token));
            } else {
                int b = stack.pop();
                int a = stack.pop();
                switch (token) {
                    case "+":
                        stack.push(a + b);
                        break;
                    case "-":
                        stack.push(a - b);
                        break;
                    case "*":
                        stack.push(a * b);
                        break;
                    case "/":
                        stack.push(a / b);
                        break;
                }
            }
        }
        return stack.pop();
    }
}
```

4. Implement the `clone()` Method for the `ArrayStack` Class

Add a cloning method for an array-based stack.

```
java
Copy code
import java.util.Arrays;

public class ArrayStack<E> implements Stack<E>, Cloneable {
    private E[] data;
    private int top = -1;

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

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

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

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

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

    @Override
    public E pop() {
        if (isEmpty()) return null;
        E result = data[top];
        data[top--] = null; // Clear to let GC do its work
        return result;
    }

    @Override
    public ArrayStack<E> clone() {
        try {
            ArrayStack<E> cloned = (ArrayStack<E>) super.clone();
            cloned.data = Arrays.copyOf(this.data, this.data.length);
            return cloned;
        } catch (CloneNotSupportedException e) {
            throw new AssertionError(); // Shouldn't happen
        }
    }
}
```

Data Structure Lab7 : Stack 2022-2023

5. Input and Evaluate a Postfix Expression

Input a postfix expression and evaluate its value using the evaluator from question 3.

java

Copy code

```
import java.util.Scanner;

public class PostfixInputEvaluator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a postfix expression: ");
        String expression = scanner.nextLine();
        int result = PostfixEvaluator.evaluate(expression);
        System.out.println("Result: " + result);
        scanner.close();
    }
}
```