ICS-202 Lab-04 Report

Mohammed Busaleh - 202158210

Exercise 2:

Code:

```java
public class BalancedParentheses {
    public static void main(String[] args) {

        //Create lists of parentheses
        ArrayList<String> open = new ArrayList<>();
        open.add("(");
        open.add("[");
        open.add("{");

        ArrayList<String> close = new ArrayList<>();
        close.add(")");
        close.add("]");
        close.add("}");



        // Takes the input, store it and removes the non-parentheses characters
        Scanner input = new Scanner(System.in);
        System.out.print("Enter your expression:  ");
        StringBuilder expression = new StringBuilder(input.nextLine());
        for(int i = 0; i <expression.length(); i++){
            String letter = String.valueOf(expression.charAt(i));
            if(!(close.contains(letter) || open.contains(letter))) {
                expression.delete(i, i + 1);
                i--;
            }
        }


        // Runs the function and shows whither the input is balanced or not
        if(isBalanced(expression.toString(), open, close))
            System.out.println("The expression is balanced");
        else
            System.out.println("The expression is not balanced");

    //FIN of main
    }
```

```java
    public static boolean isBalanced(String exp, ArrayList<String> open, ArrayList<String> close){
        LabStack<String> parentheses = new LabStack<>();

        for(int i = 0; i <exp.length(); i++){ //loops throw the parentheses
            String letter = String.valueOf(exp.charAt(i));
            if(open.contains(letter)) //The encountered parentheses is open
                parentheses.push(letter);
            else if (close.contains(letter) && !parentheses.isEmpty()) {//The encountered parentheses is close and the stack is not empty
                if(!(close.indexOf(letter) == open.indexOf(parentheses.pop())))
                    return false;
            }
            else //The encountered parentheses is close and the stack is empty
                return false;
        }


        return parentheses.isEmpty(); //After the loop checks if the stack is empty
    }
```

Output:

```
Enter your expression:  (5 * [2 + 3])
The expression is balanced
```

```
Enter your expression:  5 * (2 + 3)]
The expression is not balanced
```

```
Enter your expression:  [5 + (3 + 2])
The expression is not balanced
```

Exercise 3:

Code:

```java
public class postfix {
    public static void main(String[] args) {
        //Operators list
        ArrayList<String> ops = new ArrayList<>();
        ops.add("+");
        ops.add("-");
        ops.add("*");
        ops.add("/");

        //Read the expression and store it and make an iterator
        LabStack<Integer> numsStack = new LabStack<>();
        Scanner input = new Scanner(System.in);
        int num1, num2;
        System.out.print("Write your postfix expression:  ");
        String exp = input.nextLine();
        Scanner expLit = new Scanner(exp);
        boolean isValid = true;
```

```java
    //Iterate over the expression
    while(expLit.hasNext() && isValid){
        String c = expLit.next();
        try{
            numsStack.push(Integer.valueOf(c)); //If the token is a number, store it in the stack
        }
        catch (Exception e){ //If the token is a sign, operate based on it
            try{
            num2 = numsStack.pop();
            num1 = numsStack.pop();
            switch (ops.indexOf(c)) {
                case 0 -> numsStack.push(num1 + num2);
                case 1 -> numsStack.push(num1 - num2);
                case 2 -> numsStack.push(num1 * num2);
                case 3 -> numsStack.push(num1 / num2);
            }
            }catch(Exception e1){
                System.out.println("The expression is not valid");
                isValid = false;
            }

            if(isValid)
                System.out.println("The stack currently is: " + numsStack.toString());
        }
    }
    if(isValid)
        System.out.println(exp + "  =  " + numsStack.pop()); //Show result
}
```

Output:

```
Write your postfix expression:  2 5 6 +
The stack currently is: [2, 11]
The stack currently is: [22]
2 5 6 + *  =  22
```

```
Write your postfix expression:  5 -
The expression is not valid
```

Exercise 4:

Code:

```java
public class StackReverse {
    public static void main(String[] args) {
        LabStack<String> org = new LabStack<>();
        LabQueue<String> reversed = new LabQueue<>();

        Scanner input = new Scanner(System.in);
        System.out.print("\nEnter your input:  ");
        String inp = input.nextLine();
        Scanner reader = new Scanner(inp);

        //Iterate over the input
        while(reader.hasNext()){
            org.push(reader.next());
        }
        System.out.print("\n     Your stack is:     " + org.toString());

        //Store stack elements in a queue & back to the stack
        while(!org.isEmpty()){
            reversed.enqueue(org.pop());
        }
        while(!reversed.isEmpty()){
            org.push(reversed.dequeue());
        }

        //Print reversed elements
        System.out.println("\n\nYour Reversed stack is: " + org.toString());
    }
}
```

Output:

```
Enter your input:  1 2 3

     Your stack is:     [1, 2, 3]

Your Reversed stack is: [3, 2, 1]
```