Lab 03
Data Structures
BS DS Fall 2024 Morning/Afternoon

Objective(s):

- Understanding and finding LIFO behaviour in different problems.

Instructions:

- Make sure that there are no ***dangling pointers*** or ***memory leaks*** in your programs.
- Indent your code properly.
- Use meaningful variable and function names. Follow the naming conventions.
- Use meaningful prompt lines/labels for all input/output that is done by your programs.

Task-1

A palindrome is a word, phrase, number or other sequence of units that has the property of reading the same in either direction. Write a program that could determine whether the given string is a palindrome.
Examples:

Palindrome Words:
Dad
1221
Racecar
Rotator
Level
Civic
Palindrome phrases:
Too bad--I hid a boot.
Do geese see God?
Go Hang a Salami! I'm a Lasagna Hog!
(title of a book on palindromes by Jon Agee, 1991)

Note: Handle spaces and punctuation marks carefully.

Task-2

Check whether a string is of the form a^nb^n where n = 0, 1, 2, 3, 4, …….
Examples:
aaabbb – is of the form a^nb^n
bbbaaa – is not a form of a^nb^n
ababab – is not a form of a^nb^n

Task-3

Write a program that reads from the user a mathematical expression like:

$$(3 + (4 - 9) \times (3/4) \times (3 + 2))$$

1. Your program should check if the expression is correct

An expression is correct if:
The parentheses are correctly added and are balanced.
Examples of good or bad expressions can be:

$(a \times \{b + c\} - 4/x + [e - 5])$	GOOD
$(5+\}$	BAD
$(5 + \{6 \times) - 2\}$	BAD
$(5 + z$	BAD

Task # 4

Here is the description of the problem of **Adding two very large numbers** as described in Section 4.1 of the book "Data Structure And Algorithms in C++" (4th Edition) by Adam Drozdek:

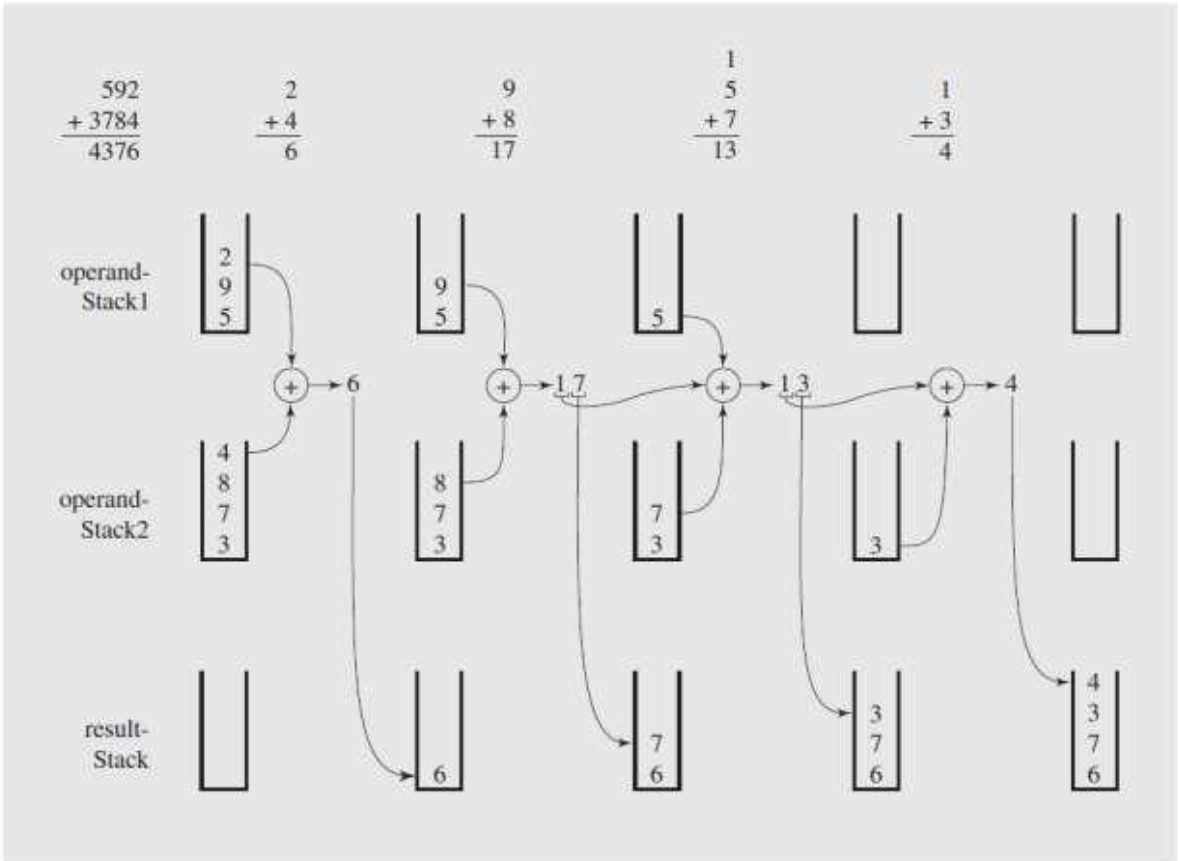
The largest magnitude of integers is limited, so we cannot add 18,274,364,583,929,273,748,459,595,684,373 and 8,129,498,165,026,350,236, because integer variables cannot hold such large values, let alone their sum. The problem can be solved if we treat these numbers as strings of numerals, store the numbers corresponding to these numerals on two stacks, and then perform addition by popping numbers from the stacks. The pseudocode for this algorithm is as follows:

addingLargeNumbers()

a. *read the numerals (digits) of the first number and store the numbers corresponding to them on one stack;*
b. *read the numerals (digits) of the second number and store the numbers corresponding to them on another stack;*
c. carry = 0;
d. while *at least one stack is not empty*
   i. *pop a number from each nonempty stack and add them to* carry*;*
   ii. *push the unit part on the result stack;*
   iii. *store carry in* carry*;*
e. *push carry on the result stack if it is not zero;*
f. *pop numbers from the result stack and display them;*

Figure 4.3 shows an example of the application of this algorithm. In this example, numbers 592 and 3,784 are added.

FIGURE 4.3 An example of adding numbers 592 and 3,784 using stacks.



The step-by-step working of the above example is explained below:

1. Numbers corresponding to digits composing the first number are pushed onto operandStack1, and numbers corresponding to the digits of 3,784 are pushed onto operandStack2. Note the order of digits on the stacks.
2. Numbers 2 and 4 are popped from the stacks, and the result, 6, is pushed onto resultStack.
3. Numbers 9 and 8 are popped from the stacks, and the unit part of their sum, 7, is pushed onto resultStack; the tens part of the result, number 1, is retained as a carry in the variable carry for subsequent addition.
4. Numbers 5 and 7 are popped from the stacks, added to the carry, and the unit part of the result, 3, is pushed onto resultStack, and the carry, 1, becomes a value of the variable carry.
5. One stack is empty, so a number is popped from the nonempty stack, added to carry, and the result is stored on resultStack.
6. Both operand stacks are empty, so the numbers from resultStack are popped and printed as the final result.

You are required to write a **C++ program** which implements the algorithm for adding two very large numbers. Here are a few instructions that you need to keep in mind:

1. Implement and use the Stack class (for storing integers) that we have seen in class.

2. Your program should take two numbers from the user and store them as two c-strings. Assume that the maximum number of numerals (digits) in a number is 25.

3. After taking input, your program should determine the sum of these two large numbers using the aforementioned algorithm and display the sum on screen.