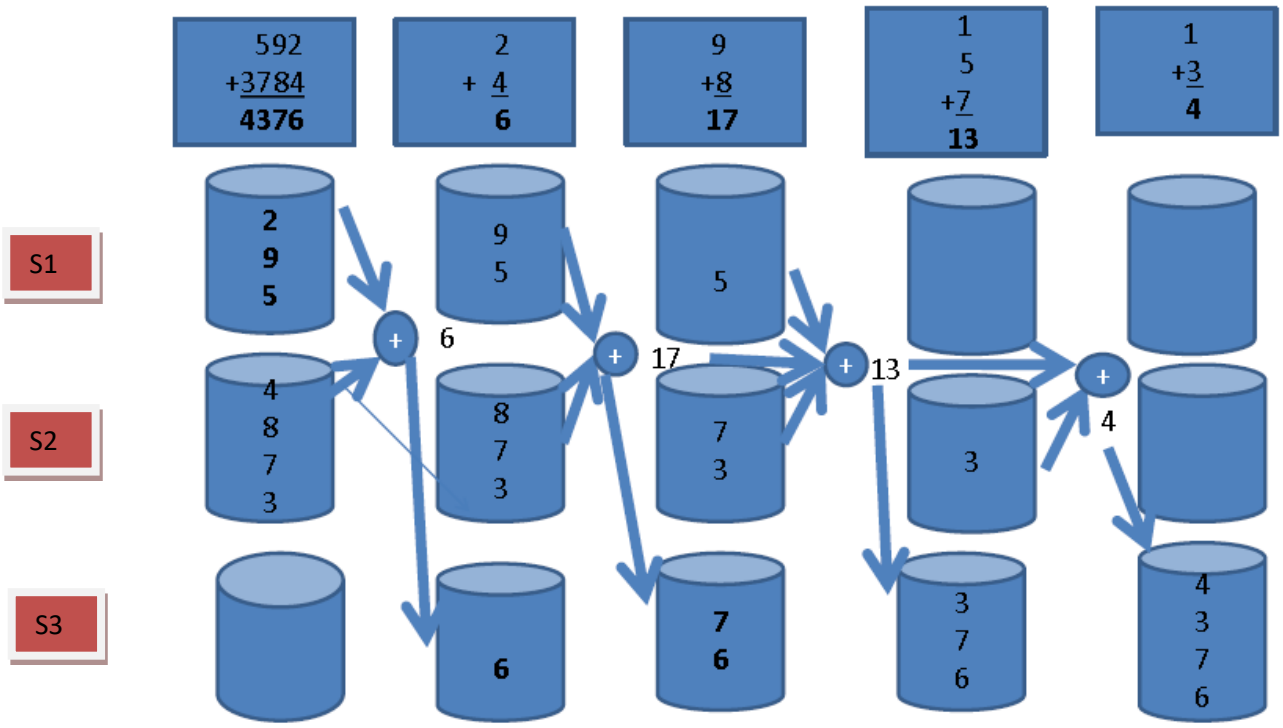
Online 2 (November 18, 2018)

Time: 40 Minutes Marks: 10

Introduction

In this assignment, you have to implement a **StackAdd** class. You need to use your offline on stack for this purpose. Your implementation must follow object oriented approach.

The objective of StackAdd class is to add two numbers stored in two separate stacks and then store the result in another stack as shown below in the figure. The numbers 592 and 3784 are stored in two stacks S1 and S2. To store one number in a stack each digit of the number is pushed in the stack separately. While adding the numbers, digits of the same position are popped from the stacks and the resultant digit is stored in the stack S3 (Don't forget to handle the carry in). The detail of the functions provided by the StackAdd class is described below.



Problem Specification

You have to design & implement a class named **StackAdd** having only **two** variables of **stack** type. **You cannot use any other variable as a class member but you can have local variables inside your member function**. The operations to be implemented are described in brief.

- **Initialization:** Write suitable constructors to initialize two stack objects with the given parameters. The parameters can be either of type integer (e.g., 592 can be used to initialize a stack as shown in the figure) or stack objects (Hints: use push and pop functions).

- **Add:** The purpose is to add two numbers stored in stacks and returns the result in a stack. The two stacks storing the numbers that will be added can be available in two ways: (i) Both of the stacks can be passed as parameters (you must use call by value method to pass the parameters), or (ii) stacks are the members of the class.
  Overload the add functions to achieve this. Note that, name of the functions must be the same for all these two ways.

- **Replace:** There should be ways to replace the numbers stored in the member stacks of the class **StackAdd**. Here, the new numbers can be supplied only through stack parameters (Hints: use push and pop functions).

Input Output Specification

In your main function, you will declare a **StackAdd** object and two **stack** objects. You have to prompt for input in a loop. Show a menu like the one shown below.

1: Input two numbers	4: Add stacks
2: Input two stacks	5: Replace
3: Add	6: Exit

Based on the response for the selection, take necessary inputs. For example, if the user chooses selections 2, 4 or 5, take $m+n+2$ numbers as input where m denotes the size of first stack and n denotes the size of second stack. Following m numbers are elements of the first stack and n numbers are elements of the second stack. For selection 1, you will take two integer numbers as input. For selection 1, 2, and 5, the inputs will be used to initialize the stack objects of **StackAdd** class.

For 3 and 4, you have to print the content of the returned stack by the add function as output.

The difference between selections 3 and 4 is that for 4, the stacks storing the numbers that will be added will be first given as inputs and passed as parameters of the add function, whereas for 3 the add function will work on the stack objects of **StackAdd** class, which already have data through the selection 1, 2 or 5.

Marks Distribution:

Will be decided by the evaluator.

Special Instructions:

* If you **have not completed Offline 2**, you can use built-in **stack** data structure from standard template library of C++ (if you can use it), to get partial marks based on what you can implement in this online.