National University of Computer and Emerging Sciences

Laboratory Manual

for

Data Structures Lab

Course Instructor	Mr. Saad Farooq
Lab Instructor(s)	Husnain Iqbal Usama Hassan
Section	CS-E
Semester	Fall 2021

Department of Computer Science

FAST-NU, Lahore, Pakistan

Objectives:

In this lab, students will practice:

1. Stack Implementation using Variable-sized Dynamic Arrays
2. Stack Implementation using Linked Lists
3. The applications of Stacks

1. Implement a template-based stack using a variable-sized dynamic array. When the array gets full, double its size before insertion of new elements. When array contains data lesser than 33% of its size, reduce its size to half. The required member methods are:
   T* arr: array containing data elements of stack
   int capacity: stores the capacity of stack**.**
   int getCout(): returns total elements stored in the stack.
   bool isEmpty(): returns true if the stack is empty else false.
   bool isFull(): returns true if the stack is full.
   bool top(): returns, but does not delete, the topmost element from the stack via the parameter passed by reference, and returns true via the return statement. If there is no element, it returns false via the return statement.
   bool pop(): deletes the top most element from the stack and returns true via the return statement. If there is no element, it returns false.
   bool push(T const& e): pushes the element "e" on top of the stack if there is some space available, and returns true via the return statement. Otherwise, If there is no capacity in the stack than call resize function to increase the capacity of stack.
   void reSize() This function will double the size of stack.
   Optional (This function should also reduce the size of stack by half if the count of elements is less than 25% of the total capacity).

2. Implement a template-based stack using linked list. Maintain the head pointer in the stack class. The required members and methods are:
   int size(): returns total elements stored in the stack
   bool isEmpty(): returns true if the stack is empty else false.
   bool top(T& data): returns, but does not delete, the topmost element from the stack via the parameter passed by reference, and returns true via the return statement. If there is no element, it returns false.
   bool pop(): deletes the top most element from the stack and returns true via the return statement. If there is no element, it returns false.
   void push(T const& e): pushes the element "e" on top of the stack.
   (optional) ReverseString (String s): this method should reverse the parameter string.

3. Given an expression containing opening and closing braces, brackets, and parentheses; implement a global function "isBalanced" to check whether the given expression is a balanced expression or not, using your stack implementation. For example, {[{}{}]}[()], {{}{}}, and []{}() are balanced expressions, but {()}[) and {(}) are not balanced. In your main function test your function using the given examples.
   `bool isBalanced(string exp)`