**University of Central Punjab**

**Faculty of Information Technology**

# Data Structures and Algorithms

# Spring 2021

|  |  |  |
| --- | --- | --- |
| **Lab 04** | |  |
| **Topic** | * Abstract Classes * Templates * GrowAble Arrays * GrowAble Stacks * Queues |
| **Objective** | The basic purpose of this lab is to implement ADT of stack, Queue and test its applications. |
|  | | |

**Instructions:**

* Indent your code.
* Comment your code.
* Use meaningful variable names.
* Plan your code carefully on a piece of paper before you implement it.
* Name of the program should be same as the task name. i.e. the first program should be Task\_1.cpp
* **void main() is not allowed. Use int main()**
* **You have to work in multiple files. i.e separate .h and .cpp files**
* **You are not allowed to use system**("**pause**")
* **You are not allowed to use any built-in functions**
* **You are required to follow the naming conventions as follow:**
  + **Variables:** firstName; (no underscores allowed)
  + **Function:** getName(); (no underscores allowed)
  + **ClassName:** BankAccount (no underscores allowed)

**Students are required to complete the following tasks in lab timings.**

## Task 1

Create a C++ generic abstract class named as **List**, with the following:

**Attributes:**

1. Type \* arr;
2. int maxSize;
3. int currentSize;

**Functions:**

virtual void addElement(Type) = 0;

* Should add the element at the last position of the **List**

virtual Type removeElement() = 0;

* Should remove the element from the last position of the **List**
* Write non-parameterized constructor for the above class.
* Write Copy constructor for the above class.
* Write Destructor for the above class.

**Remember:** Your List should not be of the fixed sized. It should be able to grow itself.

**Hint:** Write a member function regrow for this purpose. It should be private member of the class. Its prototype can be:

**void regrow(Type element)**

## Task 2

**Stack:**

Stacks are a type of container adaptors with LIFO (Last In First Out) type of working, where a new element is added at one end and (top) an element is removed from that end only.Your Stack should not be of the fixed sized. It should be able to grow itself. So using the class made in task 1, make a class named as **Stack**, having following additional functionalities:

**bool** [**empty()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) : Returns whether the **Stack** is empty or not. Time Complexity should be: O(1)

**bool** [**full()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) **:** Returns whether the **Stack** is full or not. Time Complexity should be: O(1)  
**int** [**size()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) : Returns the current size of the **Stack**. Time Complexity should be: O(1)  
**Type** [**top ()**](https://www.geeksforgeeks.org/stack-top-c-stl/) : Returns the last element of the **Stack.** Time Complexity should be: O(1)

**void** [**push(Type)**](https://www.geeksforgeeks.org/stack-push-and-pop-in-c-stl/) : Adds the element of type Type at the top of the stack. Time Complexity should be: O(1) **Hint:** You can use addElement function in it.

**Type** [**pop()**](https://www.geeksforgeeks.org/stack-push-and-pop-in-c-stl/) : Deletes the top most element of the stack and returns it. Time Complexity should be: O(1) **Hint:** You can use removeElement function in it.

* Write non-parameterized constructor for the above class.
* Write Copy constructor for the above class.
* Write Destructor for the above class.

## Task 3

Now write a global function show stack which should display all the contents of the stack.

void showStack(Stack <Type> s);

Instantiate several objects of Stack, test all the functions of Stack on them and then display them through showStack function.

## Task 4

Write a program that reads a line of text, and places each letter both in a queue and onto a stack. The program should then verify whether the line of text is a palindrome (a set of letters or numbers that is the same whether read forward or backward).

## Task 5

Write a program that sorts the values of the stack and stores them in ascending order. Create a function Display(); which will display the contents in ascending order.

## Task 6 (Class Participation Task)

Now write a program to convert an infix expression to a prefix expression.

Enter Infix expression : (A+B)\*C

Prefix Expression: +A\*BC