# 

# 

# University of Central Punjab Faculty of Information Technology

**Data Structures and Algorithms Spring 2024**

|  |  |
| --- | --- |
| **Graded Lab** |  |
| * Abstract Classes * Stacks * Stack Application |  |

**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.

|  |  |
| --- | --- |
| **Marks Division for Graded Task** | |
| **Submission:** | **Mark(s) = 1** |
| **Using proper naming conventions (functions, variables):** | **Marks = 2** |
| **Proper filing(cpp, header):** | **Marks = 2** |
| **Stack Implementation:** | **Mark(s) = 1** |
| **Implementation of ‘reverseStack’ function:** | **Marks = 2** |
| **Implementation of ‘RemoveDuplicates’ function:** | **Marks = 2** |
| **Total:** | **10 Marks** |

**Graded Task**

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

**Attributes:**

1. Type \* stackArray;
2. int maxSize;
3. int stackTop;

**Functions:**

virtual void Push(Type) = 0;

* + Should add the element at the top of stack virtual Type Pop() = 0;
  + Should remove the element from the top of stack

Using the above ‘**Stack’** class, make another derived class named as ‘**MyStack’** with the following functionalities:

**bool empty():** Returns whether the **Stack** is empty or not.

**bool full():** Returns whether the **Stack** is full or not.

**Int size():** Returns the current size of Stack.

**Type top():** Returns the last element of the **Stack.**

**void display():** Should display the stack.

* *Now create a function named* ***reverseStack*** *that takes stack as an input and returns the reversed stack as an output. The* ***reverseStack*** *function should reverse the input stack using only the isFull, isEmpty, push, and pop functions of the stack. No other function should be used.*

**Type reverseStack(Type &inputStack)**

* *Create another function named* ***RemoveDuplicates*** *that takes stack as an input. The* ***RemoveDuplicates*** *function should remove the duplicate entries from the stack and then return updated stack as an output.*

**Type RemoveDuplicates(Type &inputStack)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 

|  |  |  |
| --- | --- | --- |
| **Lab 05** | |  |
| **Topic** | * Abstract Classes * Queue * Queue Application |
| **Objective** | * The basic purpose of this lab is to implement ADT of Queue, and test its applications. |

**Task 1**

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

**Attributes:**

1. Type \* queueArray;
2. int maxSize;
3. int front, rear;

**Functions:**

virtual void enqueue(Type) = 0;

* + Should add element at the rear of queue.

virtual Type dequeue() = 0;

* + Should remove the element from the front of queue.

# Task 2

# Queue:

A linear queue is a data structure that stores the elements sequentially. It uses the FIFO (First In First Out) approach for accessing elements, where a new element is added at the rear (back) of the queue and removed from the front (other end) of queue. Use the class made in Task 1 to make a class named as **MyQueue**, having following additional functionalities:

**bool** [**empty()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) : Returns whether the **Queue** is empty or not.

**bool** [**full()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) **:** Returns whether the **Queue** is full or not.

**int** [**size()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) : Returns the current size of the **Queue**.

**Type** [**back ()**](https://www.geeksforgeeks.org/stack-top-c-stl/) : Returns the last element of the **Queue.**

# Implement both pure virtual functions enqueue () and dequeue() declared in base in myQueue

After Implementation of the functions in myQueue, create menu based program to perform the following operations

1. Press 1 to add a new element to the queue. **void enqueue(Type)**
2. Press 2 to remove and return the front element from queue. **Type dequeue()**
3. Press 3 to check if the queue is full. **bool full()**
4. Press 4 to check if the queue is empty. **bool empty()**
5. Press 5 to return the size of the queue. **int size()**
6. Press 6 to display the queue.
7. Press 7 to return the largest element in queue.
8. Press 8 to return the smallest element in queue.
9. Press 9 to display 2nd largest and 2nd smallest element of queue.
10. Press 0 to exit.
    * Write non-parameterized constructor for the above class.
    * Write Copy constructor for the above class.
    * Write Destructor for the above class.