# 

# University of Central Punjab Faculty of

# Information Technology

**Data Structures and Algorithms Spring 2024**

|  |  |
| --- | --- |
| **Graded Lab 02** |  |
| * Abstract Classes * Queue * Queue 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** |
| **Queue Implementation:** | **Mark(s) = 1** |
| **Implementation of ‘SecondLargestAndSmallest’****function:** | **Marks = 2** |
| **Implementation of ‘RotateFromFront’ function:** | **Marks = 2** |
| **Total:** | **10 Marks** |

**Graded Task**

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.

Using the above ‘**Queue’** class, make another derived class named as ‘**MyQueue’** with the following 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.**

* *Now create a function named ‘****SecondLargestAndSmallest’*** *that prints or displays one second largest and one second smallest element from the queue (you can use object of stack as parameter or without parameter it’s your own choice)*

**Type SecondLargestAndSmallest(Type &inputQueue)**

**or**

**void SecondLargestAndSmallest ()**

* *Create another function named ‘****RotateFromFront’****. This function should rotate the elements of queue from front i.e. the function should move the element at front to the rear, while the other elements will move towards front (you can use object of stack as parameter or without parameter it’s your own choice)*

**Type RotateFromFront(Type &inputQueue)**

**or**

**void RotateFromFront()**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Index** | **0** | **1** | **2** | **3** | **4** | **5** |
| **Sample Input:** | 2 | 4 | 6 | 7 | 8 | 9 |
| **Sample Output:** | 4 | 6 | 7 | 8 | 9 | 2 |