CS-202
DATA STRUCTURES

PA 1
Lists, Stacks and Queues

Due Date - 11:55pm on Wednesday 6th October 2021

Welcome to the first programming assignment of this course.

In this assignment, you are required to implement a linked list, stack, queue.

You may test the implemented data structures. However, please make sure to comment that specific part before submission. This will be an efficient way to build on your debugging skills.

For submission, please zip all files in a folder and you are required to submit the assignment using the following convention:

PA1_rollnumber.zip

Note:

The course policy about plagiarism is following:

⇒ All submissions are subjected to plagiarism detection.
The teaching staff will be using MOSS.

⇒ Please be aware of the following points:

- Students cannot copy any code from the Internet.
- Students should not share their code with other students.
- If assistance received in any part of the assignment, students should indicate that specific part of code.

⇒ All students should be able to explain any part of the code they submit, if questioned.

Part 1:

DOUBLY LINKED LIST

Welcome to InstaCart, a newly introduced in food delivery system in the country!

In this part, you will be applying all the concepts you have learned in class to implement a food delivery system. Orders for the night are being placed and it is your job to deliver them.

The basic layout of a linked list is given to you in the Order.h file. The template **OrderDetails** in Order.h represents a node in a linked list. The class **Order** implements the linked list which contains pointers to FirstOrder and LastOrder and other function declarations.

Member functions:

Write implementation for the following methods:

⇒ **Order():**

Create a function such that you can initialize an instance of an order.
(Hint: default constructor)

⇒ **Order(const Order<T>& otherOrder):**

A copy constructor, this constructor copies all elements from otherOrder to new list.

⇒ **~Order():**

Delete all orders from the memory.

⇒ **void insertFirstOrder(T item):**

Function should insert and order at the start of the linked list.

⇒ **void insertLastOrder(T item):**

Function should insert order at the end of the linked list.

⇒ **void insertAfter(T newOrder, T afterOrder):**

Function should traverse the list to find afterOrder and insert the newOrder after.

⇒ **OrderDetails<T> *getFirstOrder():**

Function should return the pointer to the head of the list.

⇒ **OrderDetails <T> *getLastOrder():**

Function should return the pointer to the tail of the list.

⇒ **OrderDetails<T> *searchForOrder(T item):**

Function should return the pointer to the item if it is in the list.
Otherwise, return null.

⇒ **void deleteOrder(T item):**

Function should find the element item and delete it from the list.

⇒ **void deleteFirstOrder():**

Function should delete the first order of the list.

⇒ **void deleteLastOrder():**

Function should delete the last order of the list.

⇒ **int numberOfOrders():**

Function should return the number of orders in the list.

*Cater to all corner cases.

*You can only make changes in Order.cpp file.

Part 2:

STACKS

Let's backtrack to prep the order!

They need your assistance to get the plates ready. The platesStack class contains Order type object.

* To complete part 2 and 3, it is mandatory you use the member functions of the Order class.

Member functions:

⇒ **platesStack():**

Implement a base constructor.

⇒ **platesStack(platesStack <T>& otherplatesStack):**

Implement a copy constructor, this constructor replicates another plate stack.

⇒ **~ platesStack():**

Remove all plates from the stack permanently.

⇒ **void pushPlate(T item):**

Function should push a plate on the top of the plates stack.

⇒ **T topPlate():**

Function should return the top plate without removing it.

⇒ **T popPlate():**

Function should return and remove the first plate from the stack.

⇒ **bool noPlates():**

Function should return true if there is no plate in the stack, otherwise false.

⇒ **int numberOfPlates():**

Function should return the number of plates in the stack.

Part 3:

QUEUES

Well it's your lucky day!

InstaCart is presenting a newly added feature in their system to avail a 30% discount called the pickUp option. This feature enables you to pre-order your food and provides an option to pick it up from the headquarters.

Member functions:

⇒ **pickUp():**

Implement a base constructor.

⇒ **pickUp(pickUp<T>& otherpickUp):**

Copy all elements of otherpickUp into the new queue.

⇒ **~pickUp():**

Delete all orders in the pick-up queue from the memory.

⇒ **void enqueue(T item):**

Function should add order to the end of the queue.

⇒ **T front():**

Function should return order at the front of pick-up queue without removing it.

⇒ **T dequeue():**

Function should return and remove order at the front of the pick-up queue.

⇒ **int lengthpickUp():**

Function should return the number of orders in the pick-up queue.

⇒ **bool isEmpty():**

Function should return true if there is no order in the queue, otherwise false.

Marks distribution:

Part	*Marks*
Doubly Linked List	60
Stack	20
Queue	20

Submission Policy:

* ZIP all the files and name the folder as PA1_rollnumber.zip

*Start the assignment early!

Happy Coding:)