CSE108 – Computer Programming Lab.

Lab 11

Due at 10am

Hand in: A student with number 20180000001 should hand in a zip file named 20180000001.zip for this lab.

PART 1. [55] FIFO (First-In-First-Out) Implementation

You are working on a project to develop a food ordering system for "GTU Restaurant." The restaurant receives a continuous stream of orders, and it is essential to keep track of these orders in a first in first out queue. You are required to implement the queue using a linked list. Here are what needs to be done:

- Define a structure named Order to hold the order details, including the customer name, order ID, items ordered, and order time.
- Implement the enqueue function, which adds a new order to the linked list (can be NULL) and returns a pointer to the root of the list. The function should take the following parameters: **Order * list enqueue(Order * queue, char * customerName, int orderID, char * items, char * orderTime).**
- Implement the dequeue function, which removes and returns the oldest order from the linked list (returning a pointer to the root of the list). The function should take the following parameters: **Order * dequeue(Order * queue).**
- Implement the display function to print the linked list on the screen. The function should take the following parameters: **void display(Order * queue).**
- Implement the updateOrder function to update the items in an existing order (defined by orderID). The function should take the following parameters: **void updateOrder(Order * queue, int orderID, char * newItems).**

Requirements:

- The linked list should be dynamically allocated and implemented using a singly linked list.
- You should not use any built-in libraries or data structures for the linked list implementation.

Example Output:

```
Enqueue Operation:
Enqueued: Order ID: 101, Customer Name: Barış Ozcan, Items: Burger, Fries, Coke
# After all enqueue operations, call display() to verify the order.

Display Orders by Order Time:
- Order ID: 101, Customer Name: Barış Ozcan, Items: Burger, Fries, Coke

Enqueue Operation:
```

```
Enqueued: Order ID: 102, Customer Name: Zehra Bilici, Items: Pizza, Salad, Ice Cream


Display Orders by Order Time:

- Order ID: 102, Customer Name: Zehra Bilici, Items: Pizza, Salad, Ice Cream

- Order ID: 101, Customer Name: Barış Ozcan, Items: Burger, Fries, Coke


Enqueue Operation:

Enqueued: Order ID: 103, Customer Name: Mehmet Burak Koca, Items: Steak, Mashed Potatoes, Salad


Display Orders by Order Time:

- Order ID: 103, Customer Name: Mehmet Burak Koca, Items: Steak, Mashed Potatoes, Salad

- Order ID: 102, Customer Name: Zehra Bilici, Items: Pizza, Salad, Ice Cream

- Order ID: 101, Customer Name: Barış Ozcan, Items: Burger, Fries, Coke


Dequeue Operation:

Dequeued Orders Older than Threshold Time:

- Order ID: 103, Customer Name: Mehmet Burak Koca, Items: Steak, Mashed Potatoes, Salad

- Order ID: 102, Customer Name: Zehra Bilici, Items: Pizza, Salad, Ice Cream


Display Orders by Order Time:

- Order ID: 101, Customer Name: Barış Ozcan, Items: Burger, Fries, Coke


Display Orders by Customer Name:

- Order ID: 101, Customer Name: Barış Ozcan, Items: Burger, Fries, Coke


Update Order:

Updated Order: Order ID: 102, Customer Name: Zehra Bilici, Items: Pizza, Salad, Coke


Display Orders by Order Time:

- Order ID: 101, Customer Name: Barış Ozcan, Items: Burger, Fries, Coke

- Order ID: 102, Customer Name: Zehra Bilici, Items: Pizza, Salad, Coke


Delete Order:

Deleted Order: Order ID: 103


Display Orders by Order Time:
```

```
- Order ID: 101, Customer Name: Barış Ozcan, Items: Burger, Fries, Coke
- Order ID: 102, Customer Name: Zehra Bilici, Items: Pizza, Salad, Coke
```

2.PART [45] LIFO (Last-In-First-Out) Implementation

You are working on a project to develop a system for storing students' exam papers in a class. The exam papers are continuously submitted to the class. When it's time to collect the papers, they should be retrieved in a Last-In-First-Out (LIFO) manner.

Task Details:

- Define a structure named ExamPaper. This structure should contain the exam paper information such as the student name, student number, and score.
- Implement the push function. This function should add a new exam paper to the stack. The function should take the following parameters: **ExamPaper * push(ExamPaper * stack, char * studentName, int studentNumber, int score).**
- Implement the pop function. This function should remove the most recently added exam paper from the stack and return it. The function should take the following parameter: **ExamPaper * pop(ExamPaper * stack).**
- Implement the isEmpty function. This function should check if the stack is empty or not. The function should take the following parameter: **int isEmpty(ExamPaper * stack).**
- Optionally, you can implement the display function, which prints the exam papers in the stack. The function should take the following parameter: **void display(ExamPaper * stack).**

Requirements:

- The stack should be dynamically created and implemented using a singly linked list.
- Standard libraries or data structures should not be used for the stack implementation.

Example Output:

```
push(stack, "Zehra Bilici", 20220000001, 90);
Exam Paper Added: Student Name: Zehra Bilici, Student Number: 20220000001, Score: 90
# After all push, call display() to see the exam papers in the stack.


push(stack, "Barış Özcan", 20190000002, 85);
Exam Paper Added: Student Name: Barış Özcan, Student Number: 20190000002, Score: 85
push(stack, "Mehmet Burak Koca", 20180000010, 95);
Exam Paper Added: Student Name: Mehmet Burak Koca, Student Number: 20180000010, Score: 95


pop(stack);
```

```
Last Added Exam Paper:
- Student Name: Mehmet Burak Koca, Student Number: 20180000010, Score: 95
# After all pop operation, call display() to see the updated exam papers in the stack.

isEmpty(stack);
Stack is not empty.

display(stack);
Exam Papers in the Stack:
- Zehra Bilici, Student Number: 20220000001, Score: 90
- Barış Özcan, Student Number: 20190000002, Score: 85
```

General Rules:

1. You will have two hours to provide a solution to the given problem set.
2. You will be able to hand in your solutions via Teams in the next two hours. The submission will be closed exactly at 10am.
3. There will be an interview session immediately after the submission deadline. Starting at 10am, you will be randomly invited to attend a meeting by a TA to demonstrate your solution and answer any questions asked by the TA.
4. You must be available until 1pm to respond to the demo invitation whenever you receive it. You will have 3 minutes after you are called via Teams. If you do not answer/appear in 3 minutes, you will miss you interview.
5. If you miss your interview or are unable to give satisfactory answers to the questions, you will receive a zero for that lab even if you have submitted your solution.
6. If you have not submitted a solution in time, you will not be invited for the interview and receive zero for that lab.
7. Due to time constraints, some students may not be invited to an interview. In that case, their solutions will be graded offline.
8. Unless you aren't declared for a specific prototype, you may use arbitrary but proper function and variable names that evoke its functionality.
9. The solution must be developed on given version of OS and must be compiled with GCC compiler, any problem which rises due to using another OS or compiler won't be tolerated.
10. Note that if any part of your program is not working as expected, then you can get zero from the related part, even it is working partially.
11. Zip your solution file before uploading it to MS Teams. The zip file must contain the C file with your solution and screenshots of the valid outputs of the program.