Midterm Exam Problem Statement

Scenario: **Hospital Emergency Queue System**

By Muhammad Hammad Shakeel From Bsai

Midterm Exam – Problem Statement
Scenario: Hospital Emergency Queue System

A small hospital is managing patients in the Emergency Room (ER). Because patient priority can change quickly (new critical patients arrive, discharged patients leave, etc.), the hospital wants a flexible system to keep track of the current patients in the ER.You are required to design and manipulate this ER Queue using a **Doubly Linked List**.

Each node in the list represents one patient and stores:

- **patientID** (integer)
- Pointer to **previous** patient (prev)
- Pointer to **next** patient (next)

The ER Queue must support the following operations:

1️⃣ Add a new patient at the beginning

- Used for critical ambulance arrivals → treated first
- *(Insert from beginning)*

2️⃣ Add a new patient at the end

- Used for normal walk-in patients
- *(Insert from end)*

3️⃣ Add a new patient at a specific position

- Insert at exact position k (1-based indexing)
- Position Rules:
    - Position 1 → treat immediately
    - Position 2 → treat after first patient
- Example: Insert at position 3 → after 2nd, before 3rd
- *(Insert at specific position)*

4️⃣ Remove a patient from the beginning

- First patient has been treated and leaves
- *(Delete from beginning)*

Q1 — Implementation / Logic

Write logic in **C++** or detailed **pseudocode** for:

- insertAtBeginning(patientID)
- insertAtEnd(patientID)
- insertAtPosition(patientID, position)
- deleteFromBeginning()

Must include updates to:

- head
- tail
- prev and next pointers

✅ Handle edge cases:

- Empty list insertion
- Deleting when list contains one node
- Inserting at position 1
- Inserting beyond list length (state your handling approach)

Notes on edge cases handled

- Inserting into an **empty list** correctly sets both head and tail.
- **Deleting** when only one node exists resets head = tail = nullptr.
- Inserting at **position 1** routes to insertAtBeginning.
- Inserting at a **position > current length + 1** appends at end (clearly documented in comment). If your instructor prefers “reject with error,” that’s a 1-line change.

Q2 — Dry Run / Trace

Start with an **empty** ER queue.

Perform each operation and **draw the doubly linked list after every step**:

1. insertAtEnd(101)
2. insertAtEnd(102)
3. insertAtBeginning(200) *(critical patient)*
4. insertAtPosition(150, 2)
5. deleteFromBeginning()

6. insertAtEnd(300)

Then answer:

a) PatientID at **head** = ?
b) PatientID at **tail** = ?
c) Full **forward traversal** (head → tail)
d) Full **backward traversal** (tail → head)

Step 1: insertAtEnd(101)

Step 1: insertAtEnd(101)

List:
[101]
Head = 101, Tail = 101

Step 2: insertAtEnd(102)

List:
[101] <-> [102]
(101.next → 102, 102.prev → 101)
Head = 101, Tail = 102

Step 3: insertAtBeginning(200) // critical patient

List:
[200] <-> [101] <-> [102]
(200.next → 101, 101.prev → 200)
Head = 200, Tail = 102

Step 4: insertAtPosition(150, 2) // after 1st, before previous 2nd

List:
[200] <-> [150] <-> [101] <-> [102]
(200.next → 150 → 101 → 102, and matching prev links)
Head = 200, Tail = 102

Step 5: deleteFromBeginning()

Delete head (200). New list:
[150] <-> [101] <-> [102]
Head = 150, Tail = 102

Step 6: insertAtEnd(300)

List:
[150] <-> [101] <-> [102] <-> [300]
Head = 150, Tail = 300

Answers after Step 6

- (a) Head patientID = **150**
- (b) Tail patientID = **300**
- (c) Forward (head → tail) = **150, 101, 102, 300**

- (d) Backward (tail → head) = **300, 102, 101, 150**

Q3 — Poster Design Requirements

Your poster must include:

1️⃣ **Title**
2️⃣ **Sub-Title**

- Student Name
- Roll Number

3️⃣ **Problem Statement**
4️⃣ **Proposed Solution**
5️⃣ **Graphical Representation** after each step:

- insertAtEnd(101)
- insertAtEnd(102)
- insertAtBeginning(200)
- insertAtPosition(150, 2)
- deleteFromBeginning()
- insertAtEnd(300)

✅ **GitHub Submission Checklist**

Upload the following:

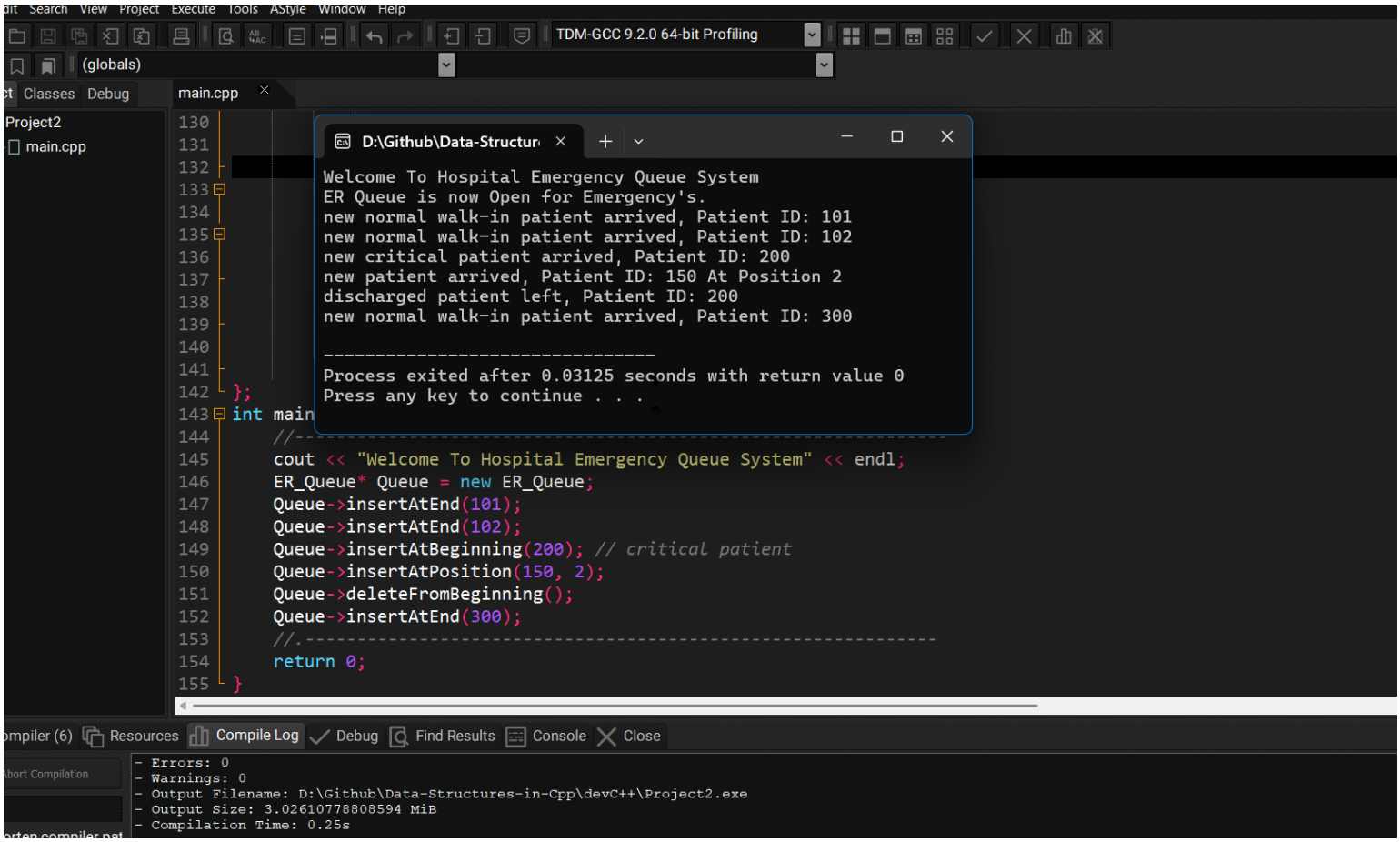
1. Complete C++ Code
2. Poster (editable format)
3. Poster in **PDF** form

```cpp
#include<iostream>
using namespace std;
/*
DLL == ER queue
    new critical patients arrive
    discharged patients leave
-----------------------------
insertAtBeginning(patientID)
insertAtEnd(patientID)
insertAtPosition(patientID, position)
deleteFromBeginning()
-----------------------------
Node == patient
    patientID (integer)
    pointers to previous patient
    pointers to next patient
-----------------------------

----------------------------
*/
class Patient {
    public:
        int patientID;
        Patient* next;
        Patient* prev;
        Patient(int patientID)
        {
            this->patientID = patientID;
            next = prev = nullptr;
        }
};
class ER_Queue{
    public:
        Patient *head;
        Patient *tail;
        ER_Queue(){
            cout << "ER Queue is now Open for Emergency's." << endl;
            head = nullptr;
            tail = nullptr;
        }
        ~ER_Queue(){
            Patient* temp = head;
            if(head != nullptr){
                cout << "Remaining Patients have been Sent Home." << endl;
            }
            while(head != nullptr){// run loop half forward from head to middle simultaneously from tail to middle delete at 2x speed
                head = head->next;
                cout << "Patient ID: " << temp->patientID << " Sent Home." << endl;
                delete temp;
                temp = head;
            }//understand this sir even though tail+prev's are dangling but after this distructor they wont exist thus it doesnt matter if i equal them to nullptr or not cos either way no one can access them anymore
            cout << "ER Queue Ended." << endl;
        }

        void insertAtBeginning(int patientID){
            cout << "new critical patient arrived, Patient ID: " << patientID << endl;
            Patient* newPatient = new Patient(patientID);
            if(head != nullptr){
                newPatient->next = head;
                head->prev = newPatient;
            }
            else{
                tail = newPatient;
            }
            head = newPatient;
        }
        void insertAtEnd(int patientID){
            cout << "new normal walk-in patient arrived, Patient ID: " << patientID << endl;
            Patient* newPatient = new Patient(patientID);
            if(tail == nullptr){//if queue empty
                head = newPatient;
                tail = newPatient;
                return;
            }
            newPatient->prev = tail;
            tail->next = newPatient;
            tail = newPatient;
        }
        void deleteFromBeginning(){
            if(head == nullptr){
                cout << "Underflow, Queue Empty." << endl;
                return;
            }
            cout << "discharged patient left, Patient ID: " << head->patientID << endl;
            Patient* temp = head;
            if(head->next != nullptr){
                head->next->prev = nullptr;
            }
            head = head->next;
            delete temp;
            if(head == nullptr){
                tail = nullptr;
            }
        }
        void insertAtPosition(int patientID, int position){
            if(position<1){
                cout << "Invalid Position, out of bounds -ve." << endl;
            }
            if(position == 1){
                insertAtBeginning(patientID);
                return;
            }
            Patient* newPatient = new Patient(patientID);
            Patient* curr = head,*temp;
            for(int i=1;i<position-1;i++){
                if(curr->next == nullptr){
                    cout << "Invalid Position, out of bounds +ve." << endl;
                    cout << "Patient Inserted At End" << endl;
                    insertAtEnd(patientID);
                    return;
                }
                curr = curr->next;
            }
            if(curr == tail){
                insertAtEnd(patientID);
                return;
            }
            temp = curr->next;
            curr->next = temp->next;
            if(temp->next != nullptr){
                temp->next->prev = curr;
            }
            delete temp;
            cout << "new patient arrived, Patient ID: " << patientID << " At Position " << position << endl;
        }
        void Display(){
            Patient* curr = head;
            if(head==nullptr){
                cout << "Queue is Empty." << endl;
                return;
            }
            while(curr != nullptr){
                cout << curr->patientID;
                if(curr->next != nullptr){
                    cout << " -> ";
                }
                curr = curr->next;
            }
            cout << endl;
        }
};
int main(){
    //-------------------------------------------------------------
    cout << "Welcome To Hospital Emergency Queue System" << endl;
    ER_Queue* Queue = new ER_Queue;
    Queue->insertAtEnd(101);
    Queue->insertAtEnd(102);
    Queue->insertAtBeginning(200); // critical patient
    Queue->insertAtPosition(150, 2);
    Queue->deleteFromBeginning();
    Queue->insertAtEnd(300);
    //.-------------------------------------------------------------
    return 0;
}

```


TDM-GCC 9.2.0 64-bit Profiling
(globals)
Classes Debug
main.cpp
Project2
main.cpp
D:\Github\Data-Structur
Welcome To Hospital Emergency Queue System
ER Queue is now Open for Emergency's.
new normal walk-in patient arrived, Patient ID: 101
new normal walk-in patient arrived, Patient ID: 102
new critical patient arrived, Patient ID: 200
new patient arrived, Patient ID: 150 At Position 2
discharged patient left, Patient ID: 200
new normal walk-in patient arrived, Patient ID: 300
Process exited after 0.03125 seconds with return value 0
Press any key to continue . . .
130
131
132
133
134
135
136
137
138
139
140
141
142 };
143 int main
144 //
145 cout << "Welcome To Hospital Emergency Queue System" << endl;
146 ER_Queue* Queue = new ER_Queue;
147 Queue->insertAtEnd(101);
148 Queue->insertAtEnd(102);
149 Queue->insertAtBeginning(200); // critical patient
150 Queue->insertAtPosition(150, 2);
151 Queue->deleteFromBeginning();
152 Queue->insertAtEnd(300);
153 //.
154 return 0;
155 }
Resources
Compile Log
Debug
Find Results
Console
Close
- Errors: 0
- Warnings: 0
- Output Filename: D:\Github\Data-Structures-in-Cpp\devC++\Project2.exe
- Output Size: 3.02610778808594 MiB
- Compilation Time: 0.25s