**Hospital Management System**

**Mahmood Ahmad Sajjad**

**Roll No : 18**

**BS AI Group B**

**Q1: Implementation Of C++ Code**

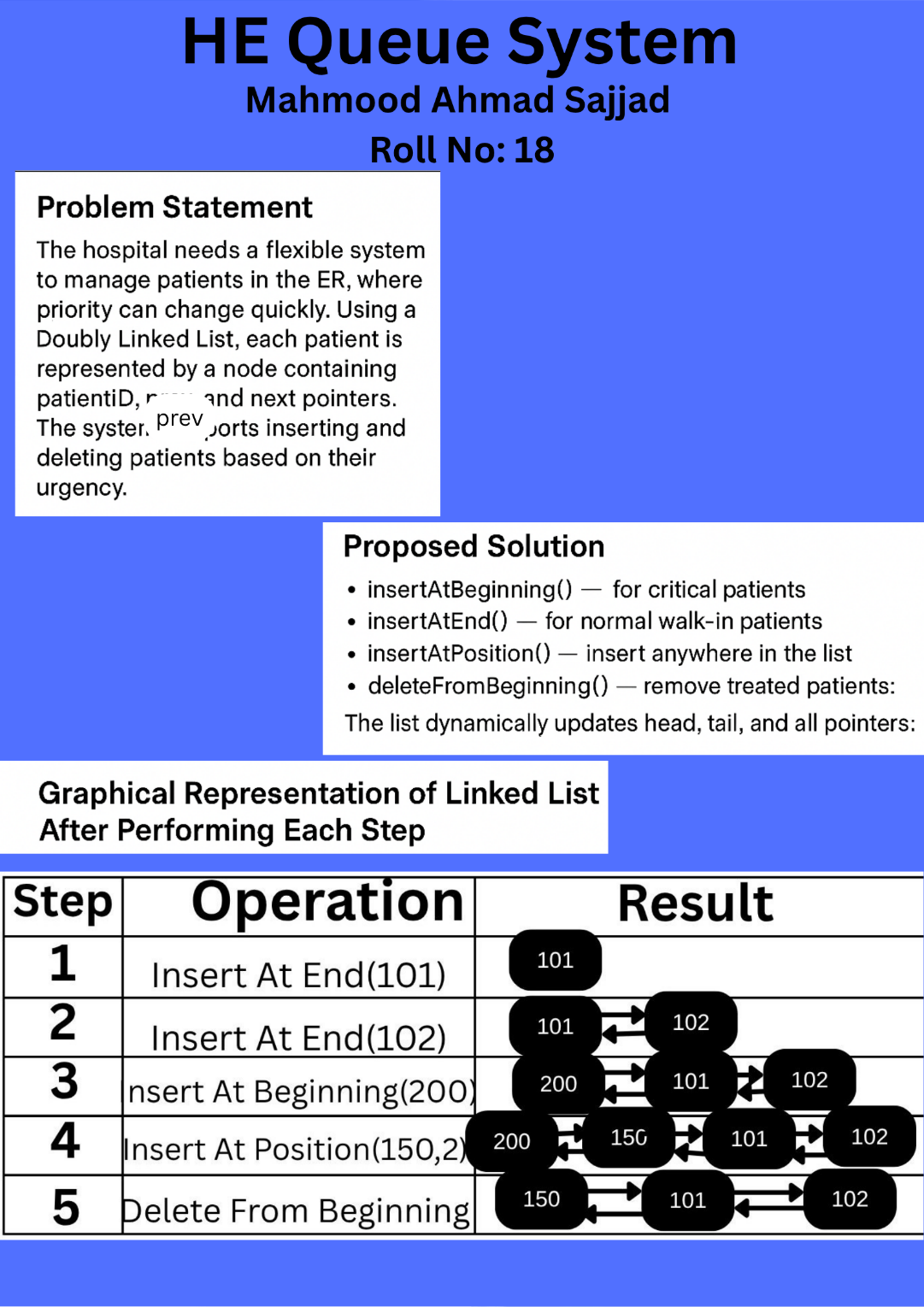
#include <iostream>  
using namespace std;  
  
class Node {  
public:  
 int patientID;  
 Node\* prev;  
 Node\* next;  
  
 Node(int id) {  
 patientID = id;  
 prev = next = NULL;  
 }  
};  
  
class ERQueue {  
 Node\* head;  
 Node\* tail;  
  
public:  
 ERQueue() {  
 head = tail = NULL;  
 }  
  
 void insertAtBeginning(int id) {  
 Node\* newNode = new Node(id);  
 if (!head) {  
 head = tail = newNode;  
 } else {  
 newNode->next = head;  
 head->prev = newNode;  
 head = newNode;  
 }  
 cout << "Critical patient " << id << " added first.\n";  
 }  
  
 void insertAtEnd(int id) {  
 Node\* newNode = new Node(id);  
 if (!head) {  
 head = tail = newNode;  
 } else {  
 tail->next = newNode;  
 newNode->prev = tail;  
 tail = newNode;  
 }  
 cout << "Patient " << id << " added at the end."<<endl;  
 }  
  
 void insertAtPosition(int id, int pos) {  
 if (pos <= 1) {  
 insertAtBeginning(id);  
 return;  
 }  
  
 Node\* newNode = new Node(id);  
 Node\* temp = head;  
 int index = 1;  
  
 while (temp && index < pos - 1) {  
 temp = temp->next;  
 index++;  
 }  
  
 if (!temp) {  
 cout << "Position too far, adding patient at the end."<<endl;  
 insertAtEnd(id);  
 return;  
 }  
  
 if (!temp->next) {  
 insertAtEnd(id);  
 } else {  
 newNode->next = temp->next;  
 newNode->prev = temp;  
 temp->next->prev = newNode;  
 temp->next = newNode;  
 cout << "Patient " << id << " placed at position " << pos << ".\n";  
 }  
 }  
  
 void deleteFromBeginning() {  
 if (!head) {  
 cout << "No patient to remove.\n";  
 return;  
 }  
  
 Node\* temp = head;  
 if (head == tail) {  
 head = tail = NULL;  
 } else {  
 head = head->next;  
 head->prev = NULL;  
 }  
  
 cout << "Patient " << temp->patientID << " treated and removed.\n";  
 delete temp;  
 }  
  
 void showForward() {  
 Node\* temp = head;  
 cout << "Queue (Front to End): ";  
 while (temp) {  
 cout << temp->patientID << " ";  
 temp = temp->next;  
 }  
 cout << endl;  
 }  
  
 void showBackward() {  
 Node\* temp = tail;  
 cout << "Queue (End to Front): ";  
 while (temp) {  
 cout << temp->patientID << " ";  
 temp = temp->prev;  
 }  
 cout << endl;  
 }  
};  
  
int main() {  
 ERQueue q;  
  
 q.insertAtEnd(101);  
 q.insertAtEnd(102);  
 q.insertAtBeginning(200);  
 q.insertAtPosition(150, 2);  
 q.deleteFromBeginning();  
 q.insertAtEnd(300);  
  
 cout << "\nFinal Queue:\n";  
 q.showForward();  
 q.showBackward();  
  
 return 0;  
}

**Q2: Dry Run**

We now perform the following operations **step-by-step**:

1. insertAtEnd(101)  
   → **List:** 101  
   (head = 101, tail = 101)
2. insertAtEnd(102)  
   → **List:** 101 ⇄ 102  
   (head = 101, tail = 102)
3. insertAtBeginning(200)  
   → **List:** 200 ⇄ 101 ⇄ 102  
   (head = 200, tail = 102)
4. insertAtPosition(150, 2)  
   → **List:** 200 ⇄ 150 ⇄ 101 ⇄ 102  
   (head = 200, tail = 102)
5. deleteFromBeginning()  
   → Removes 200  
   → **List:** 150 ⇄ 101 ⇄ 102  
   (head = 150, tail = 102)
6. insertAtEnd(300)  
   → **List:** 150 ⇄ 101 ⇄ 102 ⇄ 300  
   (head = 150, tail = 300

**Q3: POSTER**

****