ASSIGNMENT NO.3

NAME: Payal Bhagvan Charvande

CLASS: S.Y. **DIV:** IT A

PRN NO: 125B2F002

A)

CODE:-

```
#include <iostream>
using namespace std;

struct Patient {
    string name;
    int age;
    string condition;
    Patient* next;
};
class EmergencyQueue {
    Patient* head;
    public:
    EmergencyQueue() {
        head = nullptr;
    }
    // 1. Add patient
    void addPatient(string name, int age, string condition) {
        Patient* newPatient = new Patient{name, age, condition, nullptr};

        if (head == nullptr) {
            head = newPatient;
```

```
    } else {
        Patient* temp = head;
        while (temp->next != nullptr)
            temp = temp->next;

        temp->next = newPatient;
    }

    cout << "Patient added: " << name << endl;
}
// 2. Remove patient from front
void removePatient() {
    if (head == nullptr) {
        cout << "No patients in the queue.\n";
        return;
    }
    Patient* temp = head;
    head = head->next;
    cout << "Patient removed: " << temp->name << endl;
    delete temp;
}
// 3. Move a patient to the front
void moveToFront(string name) {
    if (head == nullptr || head->next == nullptr) {
        cout << "Not enough patients to move.\n";
        return;
    }
    if (head->name == name) {
        cout << "Patient is already at the front.\n";
```

```
        return;
    }

    Patient* prev = head;
    Patient* curr = head->next;

    while (curr != nullptr && curr->name != name) {
        prev = curr;
        curr = curr->next;
    }
    if (curr == nullptr) {
        cout << "Patient not found.\n";
        return;
    }
    prev->next = curr->next;
    curr->next = head;
    head = curr;

    cout << "Patient moved to front: " << name << endl;
}
// 4. Show all patients
void showQueue() {
    if (head == nullptr) {
        cout << "Queue is empty.\n";
        return;
    }
    Patient* temp = head;
    cout << "\n--- Current Queue ---\n";
    while (temp != nullptr) {
```

```
        cout << "Name: " << temp->name
             << ", Age: " << temp->age
             << ", Condition: " << temp->condition << endl;
        temp = temp->next;
    }
}
// 5. Search for a patient
void searchPatient(string name) {
    Patient* temp = head;
    while (temp != nullptr) {
        if (temp->name == name) {
            cout << "Found: " << temp->name << ", Age: " << temp->age
                 << ", Condition: " << temp->condition << endl;
            return;
        }
        temp = temp->next;
    }
    cout << "Patient not found.\n";
}
// 6. Update patient info
void updatePatient(string name, int newAge, string newCondition) {
    Patient* temp = head;
    while (temp != nullptr) {
        if (temp->name == name) {
            temp->age = newAge;
            temp->condition = newCondition;
            cout << "Updated info for: " << name << endl;
            return;
        }
```

```
            temp = temp->next;
        }
        cout << "Patient not found.\n";
    }
    // Destructor to delete all nodes
    ~EmergencyQueue() {
        while (head != nullptr) {
            Patient* temp = head;
            head = head->next;
            delete temp;
        }
    }
};
// Main program
int main() {
    EmergencyQueue queue;
    int choice;
    string name, condition;
    int age;
    do {
        cout << "\n--- Emergency Room Menu ---\n";
        cout << "1. Add Patient\n";
        cout << "2. Remove Patient (Doctor)\n";
        cout << "3. Move Patient to Front (Nurse)\n";
        cout << "4. Show Queue\n";
        cout << "5. Search Patient\n";
        cout << "6. Update Patient Info\n";
        cout << "7. Exit\n";
        cout << "Choose an option: ";
```

```
cin >> choice;


switch (choice) {
    case 1:
        cout << "Enter name (no spaces): ";
        cin >> name;
        cout << "Enter age: ";
        cin >> age;
        cout << "Enter condition (no spaces): ";
        cin >> condition;
        queue.addPatient(name, age, condition);
        break
    case 2:
        queue.removePatient();
        break;
    case 3:
        cout << "Enter name to move to front: ";
        cin >> name;
        queue.moveToFront(name);
        break;
    case 4:
        queue.showQueue();
        break
    case 5:
        cout << "Enter name to search: ";
        cin >> name;
        queue.searchPatient(name);
        break;
```

```
            case 6:
                cout << "Enter name to update: ";
                cin >> name;
                cout << "Enter new age: ";
                cin >> age;
                cout << "Enter new condition (no spaces): ";
                cin >> condition;
                queue.updatePatient(name, age, condition);
                break;
            case 7:
                cout << "Exiting...\n";
                break;
            default:
                cout << "Invalid choice.\n";
        }
    }
while (choice != 7);
    return 0;
}
```

OUTPUT:-

```
--- Emergency Room Menu ---
1. Add Patient
2. Remove Patient
3. Move Patient to Front
4. Show Queue
5. Search Patient
6. Update Patient Info
7. Exit
8. Add Multiple Patients
Choose an option: 1
Enter name: Ali
Enter age: 18
Enter condition: fever
Patient added: Ali

--- Emergency Room Menu ---
1. Add Patient
2. Remove Patient
3. Move Patient to Front
4. Show Queue
5. Search Patient
6. Update Patient Info
7. Exit
8. Add Multiple Patients
Choose an option: 2
Patient removed: Ali

--- Emergency Room Menu ---
1. Add Patient
2. Remove Patient
3. Move Patient to Front
4. Show Queue
5. Search Patient
6. Update Patient Info
7. Exit
8. Add Multiple Patients
Choose an option: 7
Exiting...
```