/\*

Problem Statement: Consider a scenario for Hospital to cater services to different kinds of patients as Serious (top priority), b) non-serious (medium priority), c) General Checkup (Least priority). Implement the priority queue to cater services to the patients.

\*/

#include <iostream>

#include <queue>

#include <vector>

#include <algorithm>

using namespace std;

// Patient structure

struct Patient {

string name;

int priority; // 1 = Serious, 2 = Non-serious, 3 = General

Patient(string n, int p) : name(n), priority(p) {}

};

// Custom comparator for sorting patients by priority

bool compare(Patient a, Patient b) {

return a.priority < b.priority; // Lower number = higher priority

}

class HospitalQueue {

vector<Patient> patients;

public:

void addPatient(string name, int priority) {

patients.push\_back(Patient(name, priority));

sort(patients.begin(), patients.end(), compare); // Keep queue sorted by priority

}

void servePatient() {

if (patients.empty()) {

cout << "No patients in the queue.\n";

return;

}

Patient p = patients.front();

patients.erase(patients.begin());

cout << "Serving patient: " << p.name << " (Priority " << p.priority << ")\n";

}

void displayQueue() {

if (patients.empty()) {

cout << "No patients in the queue.\n";

return;

}

cout << "\n--- Patient Queue (Priority Order) ---\n";

for (Patient p : patients) {

string type;

if (p.priority == 1) type = "Serious";

else if (p.priority == 2) type = "Non-Serious";

else type = "General Checkup";

cout << p.name << " - " << type << " (Priority " << p.priority << ")\n";

}

}

};

int main() {

HospitalQueue hq;

int choice;

string name;

int priority;

do {

cout << "\n--- Hospital Patient Queue ---\n";

cout << "1. Add Patient\n2. Serve Patient\n3. Display Queue\n4. Exit\nEnter choice: ";

cin >> choice;

cin.ignore();

switch (choice) {

case 1:

cout << "Enter patient name: ";

getline(cin, name);

cout << "Select patient condition:\n";

cout << "1. Serious\n2. Non-Serious\n3. General Checkup\nEnter priority (1-3): ";

cin >> priority;

if (priority >= 1 && priority <= 3)

hq.addPatient(name, priority);

else

cout << "Invalid priority!\n";

break;

case 2:

hq.servePatient();

break;

case 3:

hq.displayQueue();

break;

case 4:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice.\n";

}

} while (choice != 4);

return

Output:

--- MAIN MENU ---

1 -> Add patient

2 -> Remove patient

3 -> Get all patients

4 -> Exit

Choose an option (1-4): 1

Patient name is: John Doe

Enter priority (3-High, 2-Medium, 1-General): 3

--- MAIN MENU ---

1 -> Add patient

2 -> Remove patient

3 -> Get all patients

0 -> Exit

Choose an option (0-3): 1

Patient name is: Jane Smith

Enter priority (3-High, 2-Medium, 1-General): 2

--- MAIN MENU ---

1 -> Add patient

2 -> Remove patient

3 -> Get all patients

0 -> Exit

Choose an option (0-3): 1

Patient name is: Mike Johnson

Enter priority (3-High, 2-Medium, 1-General): 1

--- MAIN MENU ---

1 -> Add patient

2 -> Remove patient

3 -> Get all patients

0 -> Exit

Choose an option (0-3): 3

John Doe with priority: Serious patient

Jane Smith with priority: Not serious patient

Mike Johnson with priority: General checkup

--- MAIN MENU ---

1 -> Add patient

2 -> Remove patient

3 -> Get all patients

0 -> Exit

Choose an option (0-3): 2

--- MAIN MENU ---

1 -> Add patient

2 -> Remove patient

3 -> Get all patients

0 -> Exit

Choose an option (0-3): 3

Jane Smith with priority: Not serious patient

Mike Johnson with priority: General checkup

--- MAIN MENU ---

1 -> Add patient

2 -> Remove patient

3 -> Get all patients

0 -> Exit

Choose an option (1-4): 4

Exiting the program.

// END OF CODE