**Roll No. COA244**

**Assignment no .10**

INPUT :

#include <iostream> using namespace std; class Node { public: Node \*next; int priority; string data;

Node(string d, int prior) { priority = prior; data = d;

next = NULL;

}

};

class PriorityQueue { public:

Node \*front = NULL;

// Insert patient in priority order void insert(string d, int prior) {

Node \*temp = new Node(d, prior);

// If queue is empty OR new node has higher priority than front if (front == NULL || front->priority < prior) { temp->next = front; front = temp;

} else {

Node \*rear = front;

while (rear->next != NULL && rear->next->priority >= prior) { rear = rear->next;

}

temp->next = rear->next;

rear->next = temp;

}

}

// Peek first patient void peek() { if (front == NULL) { cout << "Queue is empty!" << endl;

return;

}

cout << "First patient is: " << front->data << endl;

}

// Remove the highest priority patient void pop() { if (front == NULL) {

cout << "No patients to remove!" << endl; return;

}

Node \*temp = front; front = front->next;

delete temp; // Prevent memory leak

}

// Display all patients void dis() { if (front == NULL) { cout << "Empty queue." << endl;

return;

}

cout << "\nPatient List:\n"; Node \*curr = front; while (curr != NULL) { string currPrior; if (curr->priority == 3) currPrior = "Serious patient"; else if (curr->priority == 2)

currPrior = "Not serious patient"; else

currPrior = "General checkup";

cout << curr->data << " with priority: " << currPrior << endl; curr = curr->next;

}

}

};

int main() { string name; int priority, ch; PriorityQueue q;

do {

cout << "\n--- MAIN MENU ---"; cout << "\n1 -> Add patient"; cout << "\n2 -> Remove patient"; cout << "\n3 -> Get all patients"; cout << "\n0 -> Exit"; cout << "\nChoose an option (0-3): "; cin >> ch;

switch (ch) { case 1:

cout << "Patient name: ";

cin.ignore(); getline(cin, name);

cout << "Enter priority (3-High, 2-Medium, 1-General): "; cin >> priority;

if (priority < 1 || priority > 3) {

cout << "Invalid priority! Enter between 1-3.\n"; break;

}

q.insert(name, priority); break;

case 2:

q.pop(); break;

case 3:

q.dis(); break;

case 0:

cout << "\n// END OF CODE\n";

exit(0);

default:

cout << "Invalid choice! Try again.\n";

}

} while (ch != 0);

}

OUTPUT :



