**24F-0040**

**Laiba**

**Weekly task Lab 07**

**Task from weekly task file:**

#include<iostream>

#include<string>

using namespace std;

class Queue {

public:

string data;

Queue\* face;

Queue\* rear;

Queue\* next;

string id;

string name;

int priority;

Queue() {

face = nullptr;

rear = nullptr;

next = nullptr;

}

Queue(string d) {

data = d;

}

void enqueue();

void dequeue();

int if\_Empty();

void peek();

void display();

~Queue() {

delete face;

delete rear;

delete next;

}

};

int Queue::if\_Empty() {

if (face == nullptr && rear == nullptr) {

return true;

}

return false;

}

void Queue::enqueue() {

string i, n;

int p;

cout << "Enter id of patient: ";

cin >> i;

cin.ignore();

cout << "Enter name of patient: ";

getline(cin, n);

cout << "Enter priority (1=Critical, 5=Mild): ";

cin >> p;

Queue\* new\_node = new Queue();

new\_node->name = n;

new\_node->id = i;

new\_node->priority = p;

new\_node->next = nullptr;

if (if\_Empty()) {

face = rear = new\_node;

}

else {

Queue\* temp = face;

Queue\* prev = nullptr;

while (temp != nullptr && temp->priority <= new\_node->priority) {

prev = temp;

temp = temp->next;

}

if (prev == nullptr) {

new\_node->next = face;

face = new\_node;

}

else {

prev->next = new\_node;

new\_node->next = temp;

if (temp == nullptr) rear = new\_node;

}

}

cout << "Patient " << new\_node->name << " (ID: " << new\_node->id << ", Priority: " << new\_node->priority << ") added.\n";

}

void Queue::dequeue() {

if (face == nullptr) {

cout << "Queue is empty!\n";

}

else {

Queue\* new\_node = face;

cout << "Treating patient: " << face->name << " (ID: " << face->id << ", Priority: " << face->priority << ")\n";

face = face->next;

if (face == nullptr) rear = nullptr;

delete new\_node;

}

}

void Queue::peek() {

if (face == nullptr) {

cout << "Queue is empty\n";

}

else {

cout << "Next patient: " << face->name << " (ID: " << face->id << ", Priority: " << face->priority << ")\n";

}

}

void Queue::display() {

Queue\* new\_node;

new\_node = face;

int c = 1;

cout << "All patients in Queue are : \n";

if (face == nullptr) {

cout << "Queue is Empty\n";

return;

}

else {

while (new\_node != nullptr) {

cout << "[" << new\_node->name << " (" << new\_node->priority << ")]";

if (new\_node->next != nullptr) cout << " -> ";

new\_node = new\_node->next;

c++;

}

}

cout << endl;

}

int main() {

Queue q;

int choice = 1;

while (choice != 0) {

cout << "\n--- Emergency Room Priority Queue ---\n";

cout << "1)Add Patient.\n2)Treat Patient.\n3)Next Patient.\n4)Check if Empty.\n5)Display Queue.\n0)End programme.\n";

cout << "Enter your choice: ";

cin >> choice;

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

switch (choice) {

case 1: {

q.enqueue();

break;

}

case 2: {

q.dequeue();

break;

}

case 3: {

q.peek();

break;

}

case 4: {

if (q.if\_Empty()) {

cout << "Queue is empty.\n";

}

else {

cout << "Queue is not empty.\n";

}

break;

}

case 5: {

q.display();

break;

}

case 0: {

return 0;

break;

}

default: {

cout << "Invalid input choice\n";

}

}

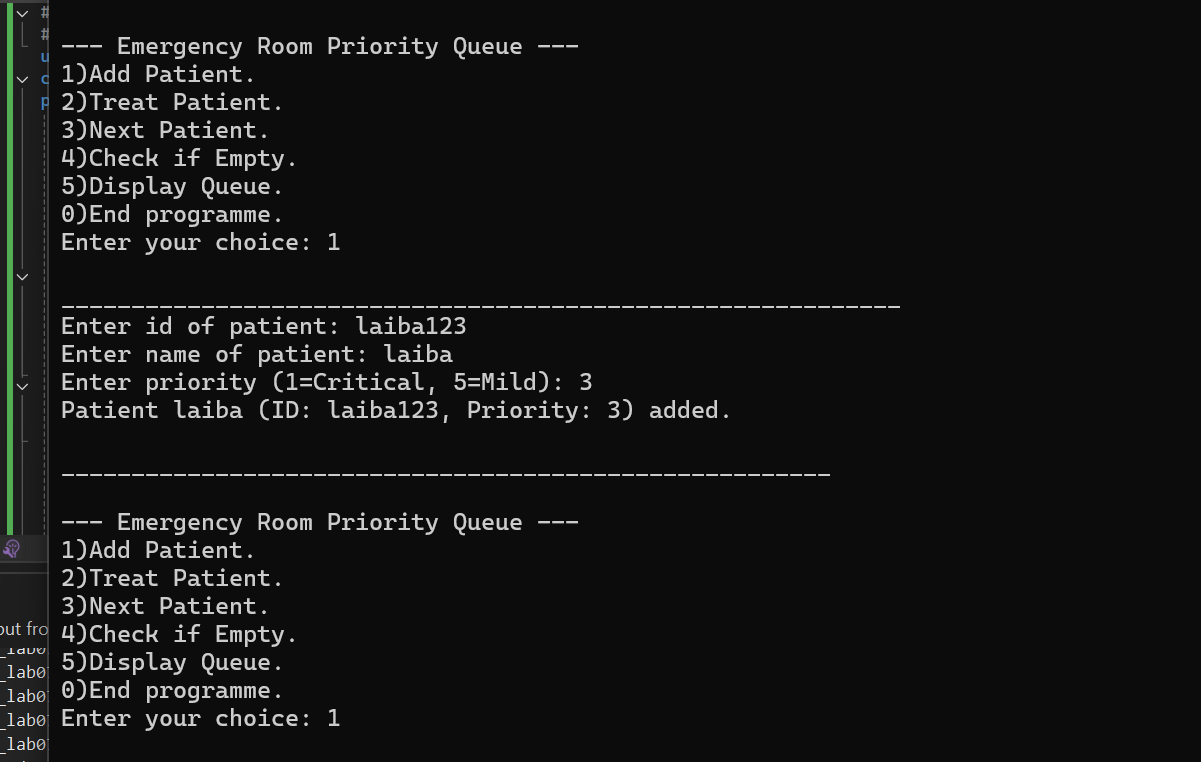
cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

}

system("Pause");

return 0;

}



A screenshot of a computer program

AI-generated content may be incorrect.

**Task 01 (Lab task given for weekly task):**

#include<iostream>

#include<string>

using namespace std;

class Queue {

public:

string data;

Queue\* face;

Queue\* rear;

Queue\* next;

string id;

string name;

int arrival;

int sizekb;

int priority;

int maxsize;

int count;

Queue() {

face = nullptr;

rear = nullptr;

next = nullptr;

count = 0;

cout << "Enter buffer size: ";

cin >> maxsize;

}

Queue(string d) {

data = d;

}

void enqueue();

void dequeue();

int if\_Empty();

void peek();

void display();

~Queue() {

delete face;

delete rear;

delete next;

}

};

int Queue::if\_Empty() {

if (face == nullptr && rear == nullptr) {

return true;

}

return false;

}

void Queue::enqueue() {

string i;

int t, s, p;

cout << "Enter Packet ID: ";

cin >> i;

cout << "Enter Arrival Time: ";

cin >> t;

cout << "Enter Size (KB): ";

cin >> s;

cout << "Enter Priority (1=Low, 2=Normal, 3=High): ";

cin >> p;

Queue\* new\_node = new Queue();

new\_node->id = i;

new\_node->arrival = t;

new\_node->sizekb = s;

new\_node->priority = p;

new\_node->next = nullptr;

if (if\_Empty()) {

face = rear = new\_node;

count++;

}

else if (count < maxsize) {

Queue\* temp = face;

Queue\* prev = nullptr;

while (temp != nullptr && temp->priority >= new\_node->priority) {

prev = temp;

temp = temp->next;

}

if (prev == nullptr) {

new\_node->next = face;

face = new\_node;

}

else {

prev->next = new\_node;

new\_node->next = temp;

if (temp == nullptr) rear = new\_node;

}

count++;

}

else {

Queue\* temp = face;

Queue\* lowest = face;

Queue\* prev = nullptr;

Queue\* prevLow = nullptr;

while (temp != nullptr) {

if (temp->priority < lowest->priority) {

lowest = temp;

prevLow = prev;

}

temp = temp->next;

if (prev == nullptr) prev = face;

else prev = prev->next;

}

if (lowest->priority < new\_node->priority) {

if (prevLow == nullptr) {

face = face->next;

}

else {

prevLow->next = lowest->next;

if (lowest == rear) rear = prevLow;

}

delete lowest;

count--;

Queue\* t2 = face;

Queue\* pr = nullptr;

while (t2 != nullptr && t2->priority >= new\_node->priority) {

pr = t2;

t2 = t2->next;

}

if (pr == nullptr) {

new\_node->next = face;

face = new\_node;

}

else {

pr->next = new\_node;

new\_node->next = t2;

if (t2 == nullptr) rear = new\_node;

}

count++;

cout << "Lowest priority packet dropped and new packet added.\n";

}

else {

cout << "Buffer full and new packet has lower or equal priority, packet dropped.\n";

delete new\_node;

}

}

}

void Queue::dequeue() {

if (face == nullptr) {

cout << "Buffer is empty!\n";

}

else {

Queue\* new\_node = face;

cout << "Forwarding packet: " << face->id << " (Priority: " << face->priority << ", Size: " << face->sizekb << "KB)\n";

face = face->next;

if (face == nullptr) rear = nullptr;

delete new\_node;

count--;

}

}

void Queue::peek() {

if (face == nullptr) {

cout << "Buffer is empty\n";

}

else {

cout << "Next packet to forward: " << face->id << " (Priority: " << face->priority << ", Size: " << face->sizekb << "KB)\n";

}

}

void Queue::display() {

Queue\* new\_node;

new\_node = face;

cout << "All packets in Buffer are : \n";

if (face == nullptr) {

cout << "Buffer is Empty\n";

return;

}

else {

while (new\_node != nullptr) {

cout << "[" << new\_node->id << " (P=" << new\_node->priority << ",T=" << new\_node->arrival << ",S=" << new\_node->sizekb << "KB)]";

if (new\_node->next != nullptr) cout << " -> ";

new\_node = new\_node->next;

}

}

cout << endl;

}

int main() {

Queue q;

int choice = 1;

while (choice != 0) {

cout << "\nEnter what do you wanna do: ";

cout << "\n1)Add Packet.\n2)Forward Packet.\n3)Next Packet.\n4)Check if Empty.\n5)Display Buffer.\n0)End programme.\n";

cin >> choice;

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

switch (choice) {

case 1: {

q.enqueue();

break;

}

case 2: {

q.dequeue();

break;

}case 3: {

q.peek();

break;

}case 4: {

if (q.if\_Empty()) {

cout << "Buffer is empty.\n";

}

else {

cout << "Buffer is not empty.\n";

}

break;

}case 5: {

q.display();

break;

}case 0: {

return 0;

break;

}default: {

cout << "Invalid input choice\n";

}

}

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

}

system("Pause");

return 0;

}

A computer screen with white text

AI-generated content may be incorrect.