National

University

of

Computer

and

Emerging

Sciences

Chiniot-Faisalabad

Campus



**CL2001 – Data Structures - Lab**

**Lab 7**

**“Queues Variations and Recursion”**

**Fall 2024**

**Maximum Marks:** 100 **Due Date:** 04 October 2024

**Submitted By**

|  |  |
| --- | --- |
| Name: | Talha Abdullah |
| Student ID: | 22F-3194 |
| Section: | BAI-3A |

**Submitted To**

Muhammad Yousaf

Lecturer – AI & Data Science Department

**Submission Date**

04 October 2024

**Table of Contents**

**Question / Title** ................................................................................................................ **Page Number**

Problem 1............................................................................................................................................... 2

Source Code........................................................................................................................................2

Screenshot..........................................................................................................................................5

Problem 2............................................................................................................................................... 5

Source Code........................................................................................................................................5

Screenshot..........................................................................................................................................5

Problem 3............................................................................................................................................... 5

Source Code........................................................................................................................................5

Screenshot........................................................................................................................................ 10

Problem 4............................................................................................................................................. 7

Source Code...................................................................................................................................... 9

Screenshot........................................................................................................................................

# **Problem 1**

## **Source Code**

#include <iostream>

using namespace std;

class Queue {

private:

struct node {

int value;

node\* next;

};

node\* front;

node\* rear;

int numItems;

public:

Queue();

~Queue();

bool isEmpty();

void makeNull();

void push(int);

bool pop(int&);

void top();

void display();

};

Queue::Queue() {

front = NULL;

rear = NULL;

numItems = 0;

}

Queue::~Queue() {

makeNull();

}

bool Queue::isEmpty() {

return front == NULL;

}

void Queue::push(int num) {

node\* newNode = new node;

newNode->value = num;

newNode->next = NULL;

if (isEmpty()) {

front = rear = newNode;

}

else if (num >= front->value) {

newNode->next = front;

front = newNode;

}

else {

node\* temp = front;

while (temp->next != NULL && temp->next->value > num) {

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

if (newNode->next == NULL) {

rear = newNode;

}

}

numItems++;

}

bool Queue::pop(int& num) {

if (isEmpty()) {

cout << "The queue is empty." << endl;

return false;

}

num = front->value;

node\* temp = front;

front = front->next;

delete temp;

if (front == NULL) {

rear = NULL;

}

numItems--;

return true;

}

void Queue::top() {

if (!isEmpty()) {

cout << "Max priority element is: " << front->value << endl;

}

else {

cout << "The queue is empty." << endl;

}

}

void Queue::display() {

if (isEmpty()) {

cout << "The queue is empty." << endl;

return;

}

node\* temp = front;

cout << "Queue elements: ";

while (temp != NULL) {

cout << temp->value << "->";

temp = temp->next;

}

cout << endl;

}

void Queue::makeNull() {

int x;

while (!isEmpty()) {

pop(x);

}

}

int main() {

Queue pq;

int choice, value;

while (true) {

cout << "------------Menu------------" << endl;

cout << "1. Push (Insert value)" << endl;

cout << "2. Pop (Remove Max priority value)" << endl;

cout << "3. Top (Display Max priority value)" << endl;

cout << "4. Display all elements in the queue" << endl;

cout << "0. Exit the program" << endl;

cout << "Enter a value to select an option: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter a value to insert: ";

cin >> value;

pq.push(value);

break;

case 2:

if (pq.pop(value)) {

cout << "Removed Max priority value: " << value << endl;

}

break;

case 3:

pq.top();

break;

case 4:

pq.display();

break;

case 0:

cout << "Exiting program..." << endl;

return 0;

default:

cout << "Invalid choice! Please try again." << endl;

break;

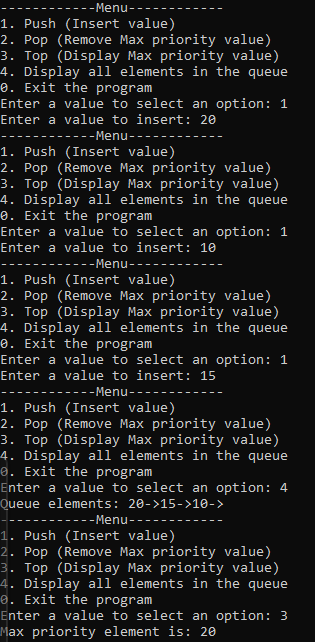
}

}

return 0;

}

## **Screenshot**



# **Problem 2**

## **Source Code**

## **Screenshot**

# **Problem 3**

## **Source Code**

#include <iostream>

#include <string>

using namespace std;

class Process {

public:

string type;

int priority;

Process(string t, int p) : type(t), priority(p) {}

};

class Queue {

private:

struct node {

Process\* process;

node\* next;

};

node\* front;

node\* rear;

int numItems;

public:

Queue();

~Queue();

bool isEmpty();

void makeNull();

void push(Process\*);

bool pop(Process\*&);

void top();

void display();

};

Queue::Queue() {

front = NULL;

rear = NULL;

numItems = 0;

}

Queue::~Queue() {

makeNull();

}

bool Queue::isEmpty() {

return front == NULL;

}

void Queue::push(Process\* process) {

node\* newNode = new node;

newNode->process = process;

newNode->next = NULL;

if (isEmpty()) {

front = rear = newNode;

}

else if (process->priority <= front->process->priority) {

newNode->next = front;

front = newNode;

}

else {

node\* temp = front;

while (temp->next != NULL && temp->next->process->priority < process->priority) {

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

if (newNode->next == NULL) {

rear = newNode;

}

}

numItems++;

}

bool Queue::pop(Process\*& process) {

if (isEmpty()) {

cout << "The queue is empty." << endl;

return false;

}

process = front->process;

node\* temp = front;

front = front->next;

delete temp;

if (front == NULL) {

rear = NULL;

}

numItems--;

return true;

}

void Queue::top() {

if (!isEmpty()) {

cout << "Process Type: " << front->process->type << ", Priority: " << front->process->priority << endl;

}

else {

cout << "The queue is empty." << endl;

}

}

void Queue::display() {

if (isEmpty()) {

cout << "The queue is empty." << endl;

return;

}

node\* temp = front;

cout << "Queue elements: ";

while (temp != NULL) {

cout << "[" << temp->process->type << ": " << temp->process->priority << "] -> ";

temp = temp->next;

}

cout << "NULL" << endl;

}

void Queue::makeNull() {

Process\* tempProcess;

while (!isEmpty()) {

pop(tempProcess);

}

}

class MultiLevelQueue {

private:

Queue systemQueue;

Queue userQueue;

public:

void push(string type, int priority);

void pop(string type);

void top(string type);

void displayAll();

};

void MultiLevelQueue::push(string type, int priority) {

Process\* newProcess = new Process(type, priority);

if (type == "System") {

systemQueue.push(newProcess);

}

else if (type == "User") {

userQueue.push(newProcess);

}

else {

cout << "Invalid process type! Please use 'System' or 'User'." << endl;

delete newProcess;

}

}

void MultiLevelQueue::pop(string type) {

Process\* process = nullptr;

if (type == "System") {

if (systemQueue.pop(process)) {

cout << "Removed process with Type: " << process->type << ", Priority: " << process->priority << endl;

delete process;

}

}

else if (type == "User") {

if (userQueue.pop(process)) {

cout << "Removed process with Type: " << process->type << ", Priority: " << process->priority << endl;

delete process;

}

}

else {

cout << "Invalid process type! Please use 'System' or 'User'." << endl;

}

}

void MultiLevelQueue::top(string type) {

if (type == "System") {

cout << "System Queue Top: ";

systemQueue.top();

}

else if (type == "User") {

cout << "User Queue Top: ";

userQueue.top();

}

else {

cout << "Invalid process type! Please use 'System' or 'User'." << endl;

}

}

void MultiLevelQueue::displayAll() {

cout << "System Queue: " << endl;

systemQueue.display();

cout << "User Queue: " << endl;

userQueue.display();

}

int main() {

MultiLevelQueue mlqArray[5];

int choice, priority, queueIndex;

string type;

while (true) {

cout << "------------Menu------------" << endl;

cout << "1. Push (Insert process)" << endl;

cout << "2. Pop (Remove process by type)" << endl;

cout << "3. Top (Display top process by type)" << endl;

cout << "4. Display all processes in a specific queue" << endl;

cout << "0. Exit the program" << endl;

cout << "Enter a value to select an option: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter queue index (0-4): ";

cin >> queueIndex;

if (queueIndex < 0 || queueIndex >= 5) {

cout << "Invalid queue index! Please enter a value between 0 and 4." << endl;

break;

}

cout << "Enter process type (System/User): ";

cin >> type;

cout << "Enter process priority: ";

cin >> priority;

mlqArray[queueIndex].push(type, priority);

break;

case 2:

cout << "Enter queue index (0-4): ";

cin >> queueIndex;

if (queueIndex < 0 || queueIndex >= 5) {

cout << "Invalid queue index! Please enter a value between 0 and 4." << endl;

break;

}

cout << "Enter process type to remove (System/User): ";

cin >> type;

mlqArray[queueIndex].pop(type);

break;

case 3:

cout << "Enter queue index (0-4): ";

cin >> queueIndex;

if (queueIndex < 0 || queueIndex >= 5) {

cout << "Invalid queue index! Please enter a value between 0 and 4." << endl;

break;

}

cout << "Enter process type to view top (System/User): ";

cin >> type;

mlqArray[queueIndex].top(type);

break;

case 4:

cout << "Enter queue index (0-4) to display: ";

cin >> queueIndex;

if (queueIndex < 0 || queueIndex >= 5) {

cout << "Invalid queue index! Please enter a value between 0 and 4." << endl;

break;

}

mlqArray[queueIndex].displayAll();

break;

case 0:

cout << "Exiting program..." << endl;

return 0;

default:

cout << "Invalid choice! Please try again." << endl;

break;

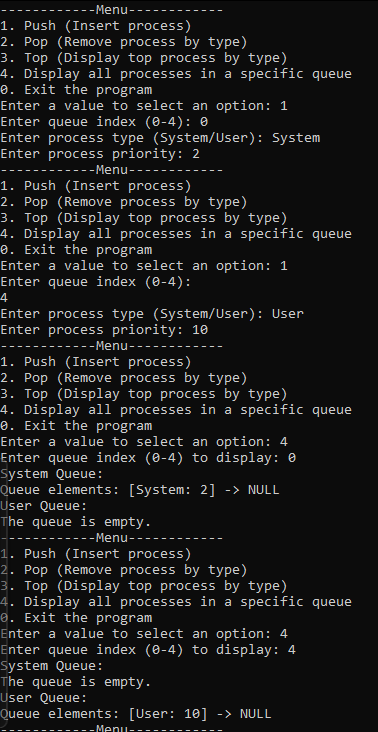
}

}

return 0;

}

## **Screenshot**



# **Problem 4**

## **Source Code**

## **Screenshot**