# Data Structures and Algorithms(UCS540)

Sixth-Semester

Submitted by:

Naman Sood [102104012] 3EE2

BE Third Year (2021-2025) Electrical Engineering

Submitted To:

Mr. Yadvendra Singh

Assistant Professor

(Contractual – I)

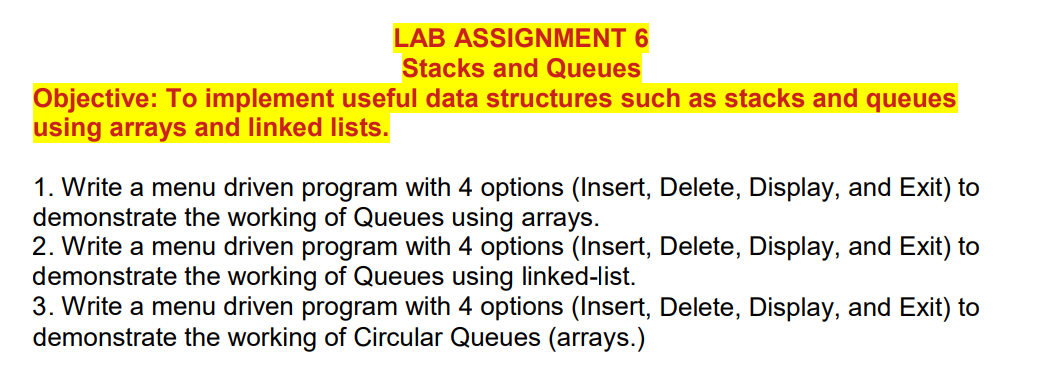


Department of Electrical & Instrumentation Engineering,

Thapar Institute of Engineering & Technology, Patiala

January-May 2024

# List of Experiments



**Q1.**

File- “QueueUsingArrays.cpp”

#include<iostream>

using namespace std;

template <typename T>

class QueueUsingArrays

{

T \*data;

int size;

int front;

int tail;

int capacity;

public:

QueueUsingArrays(int s)

{

data = new T[s];

front = -1;

tail = 0;

size = 0;

capacity = s;

}

int getSize()

{

return size;

}

bool isEmpty()

{

return size == 0;

}

void enqueue(T d)

{

if(size == capacity)

{

/\*cout<<"Queue is full!"<<endl;

return;\*/

// Dyanamic Queue

T \*newData = new T[2\*capacity];

int j = 0;

for(int i = front; i<capacity ; i++)

{

newData[j] = data[i];

j++;

}

for(int i = 0; i<capacity; i++)

{

newData[j] = data[i];

j++;

}

delete [] data;

data = newData;

front = 0;

tail = capacity;

capacity = 2\*capacity;

}

if(front == -1)

{

front = 0;

}

data[tail] = d;

tail = (tail + 1) % capacity;

size++;

}

T Front()

{

if(isEmpty())

{

cout<<"Queue is empty!"<<endl;

return 0;

}

return data[front];

}

T dequeue()

{

if(isEmpty())

{

cout<<"Queue is empty!"<<endl;

return 0;

}

T val = data[front];

data[front] = -1;

front = (front + 1) % capacity;

size--;

if(size == 0)

{

front = -1;

tail = 0;

}

return val;

}

};

File – “QueueUse.cpp”

#include <iostream>

#include "QueueUsingArrays.cpp" // Include the definition of QueueUsingArrays

using namespace std;

int main() {

int choice, size;

cout << "Enter the size of the queue: ";

cin >> size;

QueueUsingArrays<int> queue(size); // Creating a queue object of integer type

do {

cout << "\nQueue Operations Menu:" << endl;

cout << "1. Enqueue" << endl;

cout << "2. Dequeue" << endl;

cout << "3. Front" << endl;

cout << "4. Size" << endl;

cout << "5. Is Empty?" << endl;

cout << "6. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: {

int element;

cout << "Enter the element to enqueue: ";

cin >> element;

queue.enqueue(element);

cout << "Element " << element << " enqueued successfully." << endl;

break;

}

case 2: {

if (!queue.isEmpty()) {

int dequeuedElement = queue.dequeue();

cout << "Element " << dequeuedElement << " dequeued successfully." << endl;

} else {

cout << "Queue is empty. Cannot dequeue." << endl;

}

break;

}

case 3: {

if (!queue.isEmpty()) {

cout << "Front element: " << queue.Front() << endl;

} else {

cout << "Queue is empty." << endl;

}

break;

}

case 4: {

cout << "Queue size: " << queue.getSize() << endl;

break;

}

case 5: {

if (queue.isEmpty()) {

cout << "Queue is empty." << endl;

} else {

cout << "Queue is not empty." << endl;

}

break;

}

case 6: {

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

break;

}

default: {

cout << "Invalid choice. Please enter a valid option." << endl;

}

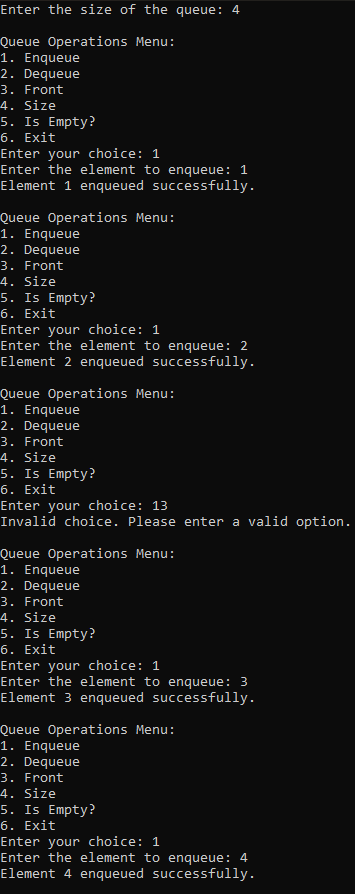
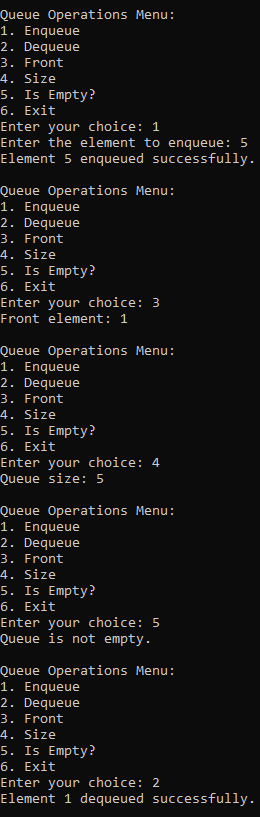
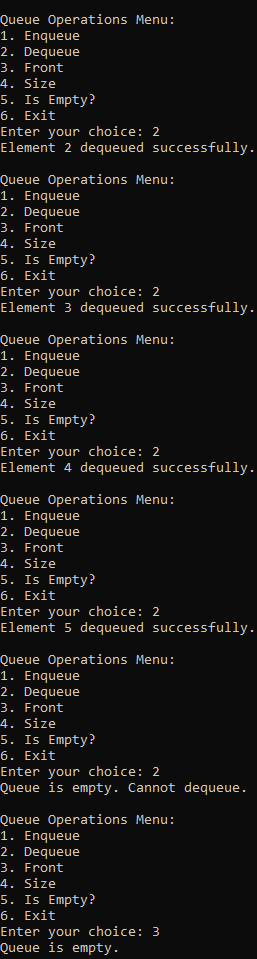
}

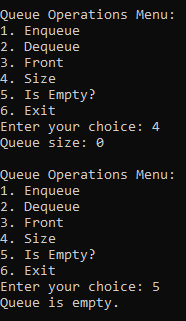
} while (choice != 6);

return 0;

}

**Output:**



**Q2.**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int value) {

data = value;

next = NULL;

}

};

class QueueUsingLinkedList {

private:

Node\* front;

Node\* rear;

public:

QueueUsingLinkedList() {

front = NULL;

rear = NULL;

}

void insert(int value) {

Node\* newNode = new Node(value);

if (front == NULL) {

front = newNode;

rear = newNode;

} else {

rear->next = newNode;

rear = newNode;

}

cout << "Element " << value << " inserted into the queue." << endl;

}

void remove() {

if (front == NULL) {

cout << "Queue is empty. Cannot delete." << endl;

return;

}

Node\* temp = front;

int deletedValue = temp->data;

front = front->next;

delete temp;

cout << "Element " << deletedValue << " deleted from the queue." << endl;

}

void display() {

if (front == NULL) {

cout << "Queue is empty." << endl;

return;

}

cout << "Queue elements: ";

Node\* current = front;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

~QueueUsingLinkedList() {

Node\* temp;

while (front != NULL) {

temp = front;

front = front->next;

delete temp;

}

}

};

int main() {

QueueUsingLinkedList queue;

int choice, element;

do {

cout << "\nQueue Operations Menu:" << endl;

cout << "1. Insert" << endl;

cout << "2. Delete" << endl;

cout << "3. Display" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter element to insert: ";

cin >> element;

queue.insert(element);

break;

case 2:

queue.remove();

break;

case 3:

queue.display();

break;

case 4:

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

break;

default:

cout << "Invalid choice. Please enter a valid option." << endl;

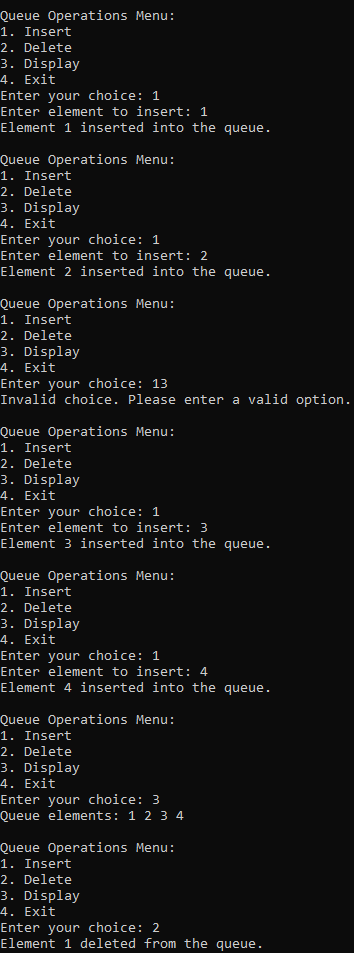
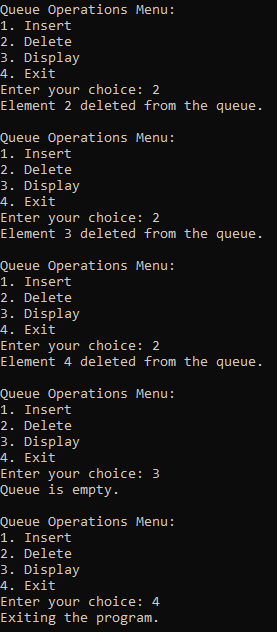
}

} while (choice != 4);

return 0;

}

**Output:**

**Q3.**

#include <iostream>

using namespace std;

#define SIZE 5 // Change the size of the queue as needed

class CircularQueue {

private:

int items[SIZE], front, rear;

public:

CircularQueue() {

front = -1;

rear = -1;

}

bool isFull() {

if (front == 0 && rear == SIZE - 1)

return true;

if (front == rear + 1)

return true;

return false;

}

bool isEmpty() {

if (front == -1)

return true;

else

return false;

}

void insertElement(int element) {

if (isFull()) {

cout << "Queue is full" << endl;

} else {

if (front == -1)

front = 0;

rear = (rear + 1) % SIZE;

items[rear] = element;

cout << "Inserted " << element << endl;

}

}

void deleteElement() {

int element;

if (isEmpty()) {

cout << "Queue is empty" << endl;

} else {

element = items[front];

if (front == rear) {

front = -1;

rear = -1;

} else {

front = (front + 1) % SIZE;

}

cout << "Deleted element: " << element << endl;

}

}

void display() {

int i;

if (isEmpty()) {

cout << "Queue is empty" << endl;

} else {

cout << "Front -> ";

for (i = front; i != rear; i = (i + 1) % SIZE)

cout << items[i] << " ";

cout << items[i] << " ";

cout << " <- Rear" << endl;

}

}

};

int main() {

CircularQueue queue;

int choice, element;

do {

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

cout << "Circular Queue Menu" << endl;

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

cout << "1. Insert" << endl;

cout << "2. Delete" << endl;

cout << "3. Display" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter element to insert: ";

cin >> element;

queue.insertElement(element);

break;

case 2:

queue.deleteElement();

break;

case 3:

queue.display();

break;

case 4:

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

break;

default:

cout << "Invalid choice, please try again!" << endl;

}

} while (choice != 4);

return 0;

}

**Output:**

