

**Data Structures**

**(Lab)**

**Task 9**

**Name:** Abdul Moiz

**SAP ID:** 54482

**Semester:** 3rd

**Question 1:**

#include<iostream>

using namespace std;

struct QueueNode {

int value;

QueueNode\* nextNode;

};

class QueueList {

QueueNode\* head;

QueueNode\* tail;

int currentSize;

int maxCapacity;

public:

QueueList(int maxCap) {

head = tail = NULL;

currentSize = 0;

maxCapacity = maxCap;

}

void addToQueue(int val) {

if (currentSize == maxCapacity) {

cout << "Queue Full. Unable to add " << val << endl;

return;

}

QueueNode\* newNode = new QueueNode;

newNode->value = val;

newNode->nextNode = NULL;

if (head == NULL) {

head = tail = newNode;

} else {

tail->nextNode = newNode;

tail = newNode;

}

currentSize++;

}

void removeFromQueue() {

if (head == NULL) {

cout << "Queue is Empty" << endl;

return;

}

QueueNode\* tempNode = head;

head = head->nextNode;

if (head == NULL) {

tail = NULL;

}

delete tempNode;

currentSize--;

}

void showQueue() {

if (head == NULL) {

cout << "Queue is Empty" << endl;

return;

}

QueueNode\* temp = head;

while (temp != NULL) {

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

temp = temp->nextNode;

}

cout << endl;

}

int queueSize() {

return currentSize;

}

};

int main() {

QueueList queue1(3);

queue1.addToQueue(10);

queue1.addToQueue(20);

queue1.addToQueue(30);

queue1.addToQueue(40);

cout << "Queue after Enqueue operations:" << endl;

queue1.showQueue();

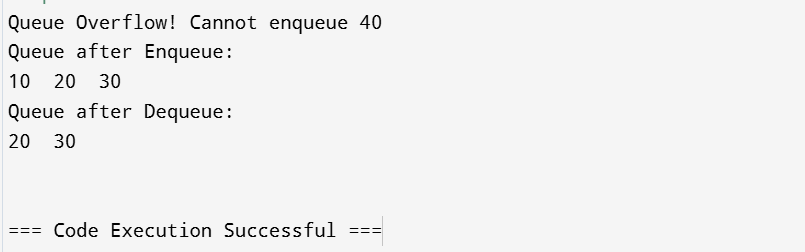
queue1.removeFromQueue();

cout << "Queue after Dequeue operation:" << endl;

queue1.showQueue();

return 0;

}



**Question 2:**

#include<iostream>

using namespace std;

struct QueueNode {

int value;

QueueNode\* nextNode;

};

class DynamicQueue {

QueueNode\* head;

QueueNode\* tail;

int currentSize;

int maxCapacity;

public:

DynamicQueue(int maxCap) {

head = tail = NULL;

currentSize = 0;

maxCapacity = maxCap;

}

void addElement(int val) {

if (currentSize == maxCapacity) {

cout << "Queue Overflow! Unable to add " << val << endl;

return;

}

QueueNode\* newNode = new QueueNode;

newNode->value = val;

newNode->nextNode = NULL;

if (head == NULL) {

head = tail = newNode;

} else {

tail->nextNode = newNode;

tail = newNode;

}

currentSize++;

}

void removeElement() {

if (head == NULL) {

cout << "Queue is Empty" << endl;

return;

}

QueueNode\* tempNode = head;

head = head->nextNode;

if (head == NULL) {

tail = NULL;

}

delete tempNode;

currentSize--;

}

void showElements() {

if (head == NULL) {

cout << "Queue is Empty" << endl;

return;

}

QueueNode\* temp = head;

while (temp != NULL) {

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

temp = temp->nextNode;

}

cout << endl;

}

int getElementCount() {

return currentSize;

}

};

int main() {

DynamicQueue myQueue(3);

myQueue.addElement(10);

myQueue.addElement(20);

myQueue.addElement(30);

myQueue.addElement(40);

cout << "Queue after adding elements:" << endl;

myQueue.showElements();

cout << "Total elements in the queue: " << myQueue.getElementCount() << endl;

myQueue.removeElement();

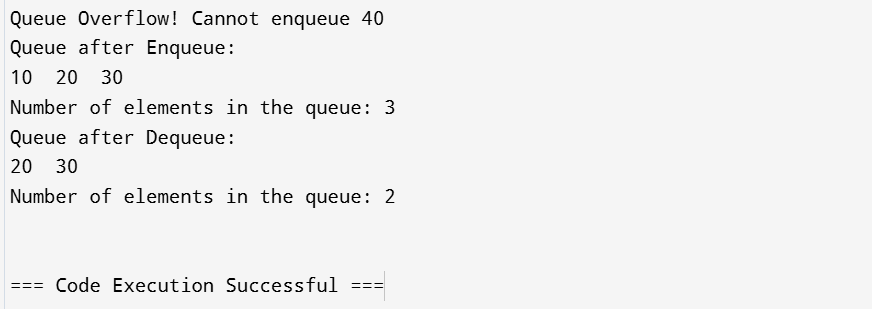
cout << "Queue after removing an element:" << endl;

myQueue.showElements();

cout << "Total elements in the queue: " << myQueue.getElementCount() << endl;

return 0;

}



**Question 3:**

#include<iostream>

using namespace std;

struct QueueNode {

int value;

QueueNode\* nextNode;

};

class CircularQueue {

QueueNode\* head;

QueueNode\* tail;

int currentSize;

int maxCapacity;

public:

CircularQueue(int maxCap) {

head = tail = NULL;

currentSize = 0;

maxCapacity = maxCap;

}

void insertElement(int val) {

if (currentSize == maxCapacity) {

cout << "Queue Overflow! Unable to insert " << val << endl;

return;

}

QueueNode\* newNode = new QueueNode;

newNode->value = val;

newNode->nextNode = NULL;

if (head == NULL) {

head = tail = newNode;

} else {

tail->nextNode = newNode;

tail = newNode;

}

currentSize++;

}

void removeElement() {

if (head == NULL) {

cout << "Queue is Empty" << endl;

return;

}

QueueNode\* tempNode = head;

head = head->nextNode;

if (head == NULL) {

tail = NULL;

}

delete tempNode;

currentSize--;

}

void clearQueue() {

while (head != NULL) {

removeElement();

}

cout << "Queue has been cleared." << endl;

}

void displayQueue() {

if (head == NULL) {

cout << "Queue is Empty" << endl;

return;

}

QueueNode\* temp = head;

while (temp != NULL) {

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

temp = temp->nextNode;

}

cout << endl;

}

int getQueueSize() {

return currentSize;

}

};

int main() {

CircularQueue myQueue(3);

myQueue.insertElement(10);

myQueue.insertElement(20);

myQueue.insertElement(30);

cout << "Queue after inserting elements:" << endl;

myQueue.displayQueue();

cout << "Total elements in the queue: " << myQueue.getQueueSize() << endl;

myQueue.clearQueue();

cout << "Total elements in the queue after clearing: " << myQueue.getQueueSize() << endl;

return 0;

}

