

**Data Structures**

**(Lab)**

**Task 9**

**Name:** Muqeet Ahmed

**SAP ID:** 53102

**Semester:** 3rd

**Question 1:**

#include<iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class Queue {

Node\* front;

Node\* rear;

int size;

int capacity;

public:

Queue(int cap) {

front = rear = NULL;

size = 0;

capacity = cap;

}

void Enqueue(int data) {

if (size == capacity) {

cout << "Queue Overflow Cannot enqueue " << data << endl;

return;

}

Node\* newnode = new Node;

newnode->data = data;

newnode->next = NULL;

if (front == NULL) {

front = rear = newnode;

} else {

rear->next = newnode;

rear = newnode;

}

size++;

}

void Dequeue() {

if (front == NULL) {

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

return;

}

Node\* temp = front;

front = front->next;

if (front == NULL) {

rear = NULL;

}

delete temp;

size--;

}

void display() {

if (front == NULL) {

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

return;

}

Node\* temp = front;

while (temp != NULL) {

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

temp = temp->next;

}

cout << endl;

}

int getSize() {

return size;

}

};

int main() {

Queue Q1(3);

Q1.Enqueue(10);

Q1.Enqueue(20);

Q1.Enqueue(30);

Q1.Enqueue(40);

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

Q1.display();

Q1.Dequeue();

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

Q1.display();

return 0;

}

**Question 2:**

#include<iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class Queue {

Node\* front;

Node\* rear;

int size;

int capacity;

public:

Queue(int cap) {

front = rear = NULL;

size = 0;

capacity = cap;

}

void Enqueue(int data) {

if (size == capacity) {

cout << "Queue Overflow! Cannot enqueue " << data << endl;

return;

}

Node\* newnode = new Node;

newnode->data = data;

newnode->next = NULL;

if (front == NULL) {

front = rear = newnode;

} else {

rear->next = newnode;

rear = newnode;

}

size++;

}

void Dequeue() {

if (front == NULL) {

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

return;

}

Node\* temp = front;

front = front->next;

if (front == NULL) {

rear = NULL;

}

delete temp;

size--;

}

void display() {

if (front == NULL) {

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

return;

}

Node\* temp = front;

while (temp != NULL) {

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

temp = temp->next;

}

cout << endl;

}

int countElements() {

return size;

}

};

int main() {

Queue Q1(3);

Q1.Enqueue(10);

Q1.Enqueue(20);

Q1.Enqueue(30);

Q1.Enqueue(40);

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

Q1.display();

cout << "Number of elements in the queue: " <<Q1.countElements() << endl;

Q1.Dequeue();

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

Q1.display();

cout << "Number of elements in the queue: " << Q1.countElements() << endl;

return 0;

}

**Question 3:**

#include<iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class Queue {

Node\* front;

Node\* rear;

int size;

int capacity;

public:

Queue(int cap) {

front = rear = NULL;

size = 0;

capacity = cap;

}

void Enqueue(int data) {

if (size == capacity) {

cout << "Queue Overflow! Cannot enqueue " << data << endl;

return;

}

Node\* newnode = new Node;

newnode->data = data;

newnode->next = NULL;

if (front == NULL) {

front = rear = newnode;

} else {

rear->next = newnode;

rear = newnode;

}

size++;

}

void Dequeue() {

if (front == NULL) {

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

return;

}

Node\* temp = front;

front = front->next;

if (front == NULL) {

rear = NULL;

}

delete temp;

size--;

}

void clear() {

while (front != NULL) {

Dequeue();

}

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

}

void display() {

if (front == NULL) {

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

return;

}

Node\* temp = front;

while (temp != NULL) {

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

temp = temp->next;

}

cout << endl;

}

int countElements() {

return size;

}

};

int main() {

Queue Q1(3);

Q1.Enqueue(10);

Q1.Enqueue(20);

Q1.Enqueue(30);

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

Q1.display();

cout << "Number of elements in the queue: " << Q1.countElements() << endl;

Q1.clear();

cout << "Number of elements in the queue after clear: " << Q1.countElements() << endl;

return 0;

}