**Name Nadeem ullah**

**Sap id 56031**

**Semester CS3-2**

**Question 1**

#include <iostream>

using namespace std;

class Queue {

private:

int front, rear, size;

int\* queue;

int capacity;

public:

Queue(int cap) {

capacity = cap;

front = rear = -1;

size = 0;

queue = new int[capacity];

}

bool isFull() {

return (size == capacity);

}

bool isEmpty() {

return (size == 0);

}

void enqueue(int value) {

if (isFull()) {

cout << "Queue Overflow! Cannot insert " << value << " as the queue is full." << endl;

return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear = (rear + 1) % capacity;

}

queue[rear] = value;

size++;

cout << "Enqueued: " << value << endl;

}

int dequeue() {

if (isEmpty()) {

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

return -1;

}

int dequeuedElement = queue[front];

front = (front + 1) % capacity;

size--;

if (isEmpty()) {

front = rear = -1;

}

cout << "Dequeued: " << dequeuedElement << endl;

return dequeuedElement;

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "Queue elements: ";

for (int i = 0; i < size; i++) {

cout << queue[(front + i) % capacity] << " ";

}

cout << endl;

}

~Queue() {

delete[] queue;

}

};

int main() {

Queue q(5);

q.enqueue(30);

q.enqueue(10);

q.enqueue(40);

q.enqueue(6);

q.enqueue(14);

q.display();

q.enqueue(60);

q.dequeue();

q.dequeue();

q.display();

q.enqueue(6);

q.display();

return 0;

}

**Question 2**

#include <iostream>

using namespace std;

class Queue {

private:

int front, rear, size;

int\* queue;

int capacity;

public:

Queue(int cap) {

capacity = cap;

front = rear = -1;

size = 0;

queue = new int[capacity];

}

bool isFull() {

return (size == capacity);

}

bool isEmpty() {

return (size == 0);

}

void enqueue(int value) {

if (isFull()) {

cout << "Queue Overflow! Cannot insert " << value << " as the queue is full." << endl;

return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear = (rear + 1) % capacity;

}

queue[rear] = value;

size++;

cout << "Enqueued: " << value << endl;

}

int dequeue() {

if (isEmpty()) {

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

return -1;

}

int dequeuedElement = queue[front];

front = (front + 1) % capacity;

size--;

if (isEmpty()) {

front = rear = -1;

}

cout << "Dequeued: " << dequeuedElement << endl;

return dequeuedElement;

}

int count() {

return size;

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "Queue elements: ";

for (int i = 0; i < size; i++) {

cout << queue[(front + i) % capacity] << " ";

}

cout << endl;

}

~Queue() {

delete[] queue;

}

};

int main() {

Queue q(5);

q.enqueue(2);

q.enqueue(21);

q.enqueue(34);

q.enqueue(5);

q.enqueue(22);

q.display();

cout << "Number of elements in the queue: " << q.count() << endl;

q.dequeue();

q.dequeue();

q.display();

cout << "Number of elements in the queue: " << q.count() << endl;

q.enqueue(6);

q.display();

cout << "Number of elements in the queue: " << q.count() << endl;

return 0;

}