**NAME : MALIHA SHAHID**

**ROLL NO:SU92-BSSEM-S24-071**

**LAB #11**

**Queue with Array & Linklist**

1. **With Array; Enqueue, Dequeue, Display**

#include <iostream>

using namespace std;

class ArrayQueue {

private:

int frontIndex, rearIndex;

int queueArray[100];

int capacity;

public:

ArrayQueue(int maxSize = 100) {

capacity = maxSize;

frontIndex = rearIndex = -1;

}

void enqueue(int value) {

if (rearIndex == capacity - 1) {

cout << "Queue Overflow\n";

return;

}

if (frontIndex == -1) frontIndex = 0;

queueArray[++rearIndex] = value;

}

void dequeue() {

if (frontIndex == -1 || frontIndex > rearIndex) {

cout << "Queue Underflow\n";

return;

}

frontIndex++;

}

void display() {

if (frontIndex == -1 || frontIndex > rearIndex) {

cout << "Queue is empty\n";

return;

}

cout << "Queue: ";

for (int i = frontIndex; i <= rearIndex; i++) {

cout << queueArray[i] << " ";

}

cout << "\n";

}

};

int main() {

ArrayQueue myQueue;

myQueue.enqueue(10);

myQueue.enqueue(20);

myQueue.enqueue(30);

myQueue.display();

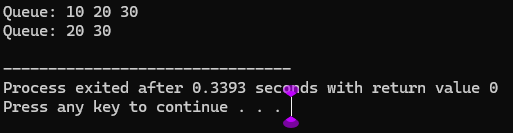
myQueue.dequeue();

myQueue.display();

return 0;

**}**

**OUTPUT**

****

**EXPLANATION**

1. This code creates a **queue using an array** called ArrayQueue, where elements follow **FIFO** (First In, First Out) order.
2. **Enqueue** adds a new element at the **rear** of the queue; it shows "Queue Overflow" if the queue is full.
3. **Dequeue** removes the element from the **front**; it shows "Queue Underflow" if the queue is empty.
4. frontIndex and rearIndex track the front and rear positions of the queue.
5. The **display** function prints all elements from front to rear.
6. In main, elements 10, 20, 30 are added, one is removed, and the queue is displayed before and after.
7. **With Linkedlist; Enqueue, Dequeue, Display**

#include <iostream>

using namespace std;

class QueueNode {

public:

int value;

QueueNode\* nextNode;

QueueNode(int val) {

value = val;

nextNode = NULL;

}

};

class LinkedQueue {

private:

QueueNode\* frontPtr;

QueueNode\* rearPtr;

public:

LinkedQueue() {

frontPtr = rearPtr = NULL;

}

void enqueue(int val) {

QueueNode\* newElement = new QueueNode(val);

if (rearPtr == NULL) {

frontPtr = rearPtr = newElement;

} else {

rearPtr->nextNode = newElement;

rearPtr = newElement;

}

}

void dequeue() {

if (frontPtr == NULL) {

cout << "Queue Underflow\n";

return;

}

QueueNode\* tempNode = frontPtr;

frontPtr = frontPtr->nextNode;

if (frontPtr == NULL) rearPtr = NULL;

delete tempNode;

}

void display() {

if (frontPtr == NULL) {

cout << "Queue is empty\n";

return;

}

cout << "Queue: ";

QueueNode\* current = frontPtr;

while (current != NULL) {

cout << current->value << " ";

current = current->nextNode;

}

cout << "\n";

}

};

int main() {

LinkedQueue myQueue;

myQueue.enqueue(100);

myQueue.enqueue(200);

myQueue.enqueue(300);

myQueue.display();

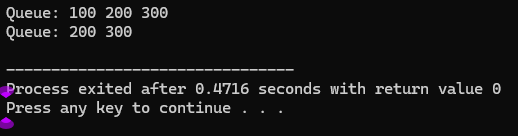
myQueue.dequeue();

myQueue.display();

return 0;

}

**OUTPUT**

****

**EXPLANATION**

1. This code creates **a queue using a linked list**, called LinkedQueue, which follows **FIFO** (First In, First Out).
2. Each element is a QueueNode that holds a value and a pointer to the next node.
3. **Enqueue** adds a new node at the **rear** of the queue; if the queue is empty, both front and rear are set to the new node.
4. **Dequeu**e removes the node from the **front**; if the queue becomes empty, rear is also set to NULL.
5. The **display** function prints all elements from front to rear.
6. In main, elements 10, 20, 30 are added, one is removed, and the queue is displayed before and after.