**Fatima Nazir 45317**

**QUEUE ARRAY**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int val)

{

data = val;

next = NULL;

}

};

class Queue

{

private:

Node\* front;

Node\* rear;

public:

Queue()

{

front = rear = NULL;

}

bool isEmpty()

{

return front == NULL;

}

bool isFull()

{

return false;

}

void enqueue(int val)

{

Node\* newNode = new Node(val);

if (rear == NULL)

{

front = rear = newNode;

} else {

rear->next = newNode;

rear = newNode;

}

cout << val << " enqueued to the queue." << endl;

}

void dequeue()

{

if (isEmpty())

{

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

return;

}

Node\* temp = front;

front = front->next;

if (front == NULL)

{

rear = NULL;

}

cout << temp->data << " dequeued from the queue." << endl;

delete temp;

}

void display()

{

if (isEmpty())

{

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

return;

}

Node\* temp = front;

cout << "Queue elements: ";

while (temp != NULL)

{

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

temp = temp->next;

}

cout << endl;

}

};

int main()

{

Queue q;

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

q.display();

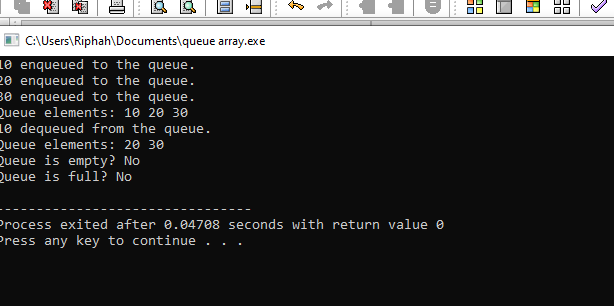
q.dequeue();

q.display();

cout << "Queue is empty? " << (q.isEmpty() ? "Yes" : "No") << endl;

cout << "Queue is full? " << (q.isFull() ? "Yes" : "No") << endl;

return 0;



**}QUEUE Link list**

#include <iostream>

using namespace std;

class Queue {

int front, rear, size;

int\* queue;

public:

Queue() {

front = rear = -1;

size = 2;

queue = new int[size];

}

~Queue() {

delete[] queue;

}

bool isFull() {

return rear == size - 1;

}

bool isEmpty() {

return front == -1 || front > rear;

}

void resize() {

int newSize = size \* 2;

int\* newQueue = new int[newSize];

for (int i = front; i <= rear; i++) {

newQueue[i - front] = queue[i];

}

delete[] queue;

queue = newQueue;

rear -= front;

front = 0;

size = newSize;

}

void enqueue(int value) {

if (isFull()) {

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

resize();

}

if (isEmpty()) {

front = rear = 0;

} else {

rear++;

}

queue[rear] = value;

cout << "Added: " << value << endl;

}

int dequeue() {

if (isEmpty()) {

cout << "Queue is empty! Cannot remove an element." << endl;

return -1;

}

int value = queue[front];

cout << "Removed: " << value << endl;

if (front == rear) {

front = rear = -1;

} else {

front++;

}

return value;

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "Queue elements: ";

for (int i = front; i <= rear; i++) {

if (queue[i] != 0) {

cout << queue[i] << " ";

}

}

cout << endl;

}

};

int main() {

Queue q;

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

q.enqueue(4);

q.display();

q.dequeue();

q.display();

q.enqueue(5);

q.enqueue(6);

q.display();

return 0;

}

