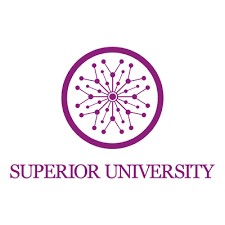
**TASK NO 11**



**M ZUHAIB ANWAR**

**SU92-BSSEM-S24-005**

**SE-3A**

**DSA LAB**

//Array (queue)

#include <iostream>

using namespace std;

const int MAX\_SIZE = 5;

class ArrayQueue

{

private:

int queue[MAX\_SIZE];

int front, rear;

public:

ArrayQueue()

{

front = -1;

rear = -1;

}

bool isFull()

{

return (front == 0 && rear == MAX\_SIZE - 1) || (rear == (front - 1) % (MAX\_SIZE - 1));

}

bool isEmpty()

{

return front == -1;

}

void enqueue(int item)

{

if (isFull())

{

cout << "It is full and cannot enqueue." << endl;

return;

}

else if (front == -1)

{

front = rear = 0;

}

else if (rear == MAX\_SIZE - 1 && front != 0)

{

rear = 0;

}

else

{

rear++;

}

queue[rear] = item;

cout << "Enqueued " << item << endl;

}

int dequeue()

{

if (isEmpty())

{

cout << "It is empty and cannot dequeue." << endl;

return -1;

}

int item = queue[front];

if (front == rear)

{

front = rear = -1;

}

else if (front == MAX\_SIZE - 1)

{

front = 0;

}

else

{

front++;

}

cout << "Dequeued " << item << endl;

return item;

}

void display()

{

if (isEmpty())

{

cout << "It is empty" << endl;

return;

}

cout << "Current Queue (front to rear): ";

if (rear >= front) {

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

{

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

}

}

else

{

for (int i = front; i < MAX\_SIZE; i++)

{

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

}

for (int i = 0; i <= rear; i++)

{

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

}

}

cout << endl;

}

};

int main()

{

ArrayQueue aq;

aq.enqueue(10);

aq.enqueue(20);

aq.enqueue(30);

aq.display();

aq.dequeue();

aq.display();

aq.enqueue(40);

aq.enqueue(50);

aq.enqueue(60);

aq.display();

aq.dequeue();

aq.dequeue();

aq.dequeue();

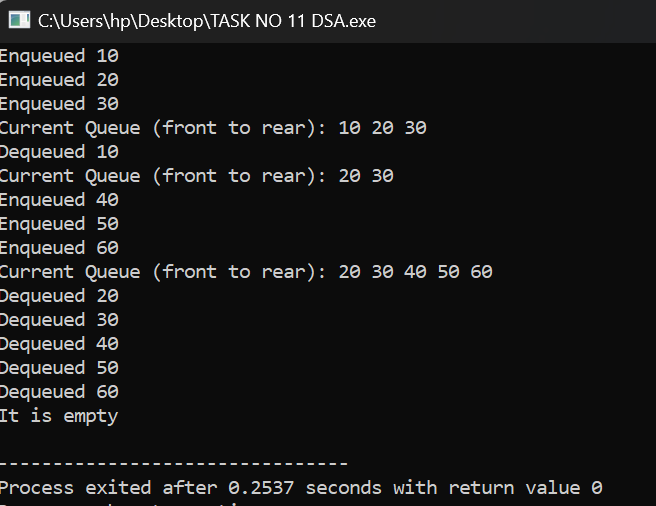
aq.dequeue();

aq.dequeue();

aq.display();

return 0;

}



//Linked list (queue)

#include <iostream>

using namespace std;

class Node

{

public:

int data;

Node\* next;

Node(int val) {

data = val;

next = nullptr;

}

};

class LinkedListQueue

{

private:

Node\* front;

Node\* rear;

public:

LinkedListQueue()

{

front = rear = nullptr;

}

bool isEmpty()

{

return front == nullptr;

}

void enqueue(int item)

{

Node\* newNode = new Node(item);

if (isEmpty())

{

front = rear = newNode;

}

else

{

rear->next = newNode;

rear = newNode;

}

cout << "Enqueued " << item << endl;

}

int dequeue()

{

if (isEmpty())

{

cout << "It is empty and cannot dequeue." << endl;

return -1;

}

Node\* temp = front;

int item = front->data;

front = front->next;

if (front == nullptr)

{

rear = nullptr;

}

delete temp;

cout << "Dequeued " << item << endl;

return item;

}

void display()

{

if (isEmpty())

{

cout << "It is empty" << endl;

return;

}

cout << "Current Queue (front to rear): ";

Node\* current = front;

while (current != nullptr)

{

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

~LinkedListQueue()

{

while (!isEmpty())

{

dequeue();

}

}

};

int main()

{

LinkedListQueue lq;

lq.enqueue(10);

lq.enqueue(20);

lq.enqueue(30);

lq.display();

lq.dequeue();

lq.display();

lq.enqueue(40);

lq.enqueue(50);

lq.enqueue(60);

lq.display();

lq.dequeue();

lq.dequeue();

lq.dequeue();

lq.dequeue();

lq.dequeue();

lq.display();

return 0;

}

