## **Queue Implementation using Linked List**

**QueueLinkedList.h (header file)**

//

// Created by f12r on ১৫/৮/২১.

//

#ifndef ASSIGNMENT\_3\_QUEUELINKEDLIST\_H

#define ASSIGNMENT\_3\_QUEUELINKEDLIST\_H

class EmptyQueue {

};

class FullQueue {

};

template<class T>

class QueueLinkedList {

struct NodeType{

T data;

NodeType \*link;

};

private:

NodeType \*front;

NodeType \*rear;

public:

QueueLinkedList();

~QueueLinkedList();

bool isEmpty();

void enQueue(T );

void DeQueue(T &);

void PrintQueue();

};

#endif //ASSIGNMENT\_3\_QUEUELINKEDLIST\_H

**QueueLinkedList.cpp (definition file)**

//

// Created by f12r on ১৫/৮/২১.

//

#include "QueueLinkedList.h"

#include <iostream>

using namespace std;

template<class T>

QueueLinkedList<T>::QueueLinkedList() {

front = NULL;

rear = NULL;

}

template<class T>

bool QueueLinkedList<T>::isEmpty() {

return front == NULL;

}

template<class T>

void QueueLinkedList<T>::enQueue(T item) {

NodeType \*newNode;

newNode = new NodeType;

newNode->data = item;

newNode->link = NULL;

if (rear == NULL) {

front = newNode;

} else {

rear->link = newNode;

}

rear = newNode;

}

template<class T>

void QueueLinkedList<T>::DeQueue(T &deItem) {

if (isEmpty()) {

throw EmptyQueue();

}

NodeType \*temp;

temp = front;

deItem = front->data;

front = front->link;

if (front == NULL) {

rear = NULL;

}

delete temp;

}

template<class T>

void QueueLinkedList<T>::PrintQueue() {

if (isEmpty()){

throw EmptyQueue();

}

NodeType \*temp;

temp = front;

while (temp!=NULL){

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

temp = temp->link ;

}

cout<<endl;

}

template<class T>

QueueLinkedList<T>::~QueueLinkedList<T>() {

NodeType \*temp;

while (front!=NULL){

temp = front;

front= front->link;

delete temp;

}

rear= NULL;

}

**main.cpp (driver file)**

#include "QueueLinkedList.h"

#include "QueueLinkedList.cpp"

#include <iostream>

using namespace std;

int main()

{

QueueLinkedList<int> list;

// check queue is empty or not

if (list.isEmpty())

{

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

}

else

{

cout << "Queue is not empty" << endl;

}

// insert item

cout<<"Enqueue 5 items: ";

for (int i = 0; i < 5; i++)

{

int x;

cin>>x;

list.enQueue(x);

}

// print the queue list

cout << "display items of the queue: ";

list.PrintQueue();

// check queue is empty or not

if (list.isEmpty())

{

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

}

else

{

cout << "Queue is not empty" << endl;

}

// dequeue item

int x;

list.DeQueue(x);

cout << "Dequeue item is: " << x << endl;

// display items

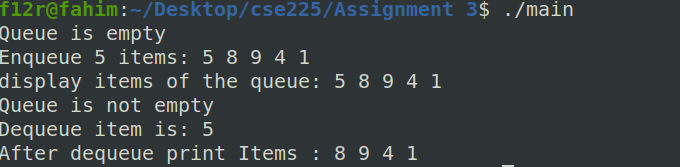
cout << "After dequeue print Items : ";

list.PrintQueue();

return 0;

}

**OUTPUT**

****