LAB WORK - 3

Aawishkar Tiwari

Computer Engineering – 3rd Semester

Roll no – 59

GitHub : https://github.com/Aawishkar/DSA_LAB.git

Queue:

queue.h

```
#ifndef Queue_h
#define Queue_h


class Queue{
    public:
        Queue(){}
        ~Queue(){}
        virtual void enqueue(int data)=0;
        virtual int dequeue()=0;
        virtual bool isEmpty()=0;
        virtual bool isFull()=0;
        virtual int  front()=0;
        virtual int  back()=0;
        virtual void display()=0;
};



#endif
```

Queue_array.h

```
#ifndef ArrayQueue_h
#define ArrayQueue_h
#define MAX_SIZE 50

#include "queue.h"


class ArrayQueue:public Queue{
    private:
        int array[MAX_SIZE];
        int fron;
        int rear;
    public:
        ArrayQueue(){
            fron=0;
            rear=0;
        }
        bool isEmpty();
        bool isFull();
        void enqueue(int data);
        int dequeue();
        int front();
        int back();
        void display();
};
#endif
```

Linkelist.h

```
#ifndef LinkedList_h
#define LinkedList_h

class Node{
    public:
        int data;
        Node *next;
        Node(){}
        Node(int x){
            data =x;
            next=nullptr;
        }
        Node(int x, Node *y){
            data =x;
            next=y;
        }
};
class LinkedList{
    public:
        Node *HEAD;
        Node *TAIL;

        LinkedList(Node *HEAD =nullptr, Node *TAIL=nullptr){
            this->HEAD=HEAD;
            this->TAIL=TAIL;
        }
        bool isEmpty();
        void addToTail(int );
        void removeFromHead();
        void traverse();
};

#endif
```

Queue_linkedlist.h

```
#ifndef LinkedlistQueue_h
#define LinkedlistQueue_h
#include "queue.h"
#include"linkedList.h"
#include <iostream>


class LinkedListQueue:public Queue{
    public:
        LinkedList list;
        LinkedListQueue();

        void enqueue(int data);
        int dequeue();
        bool isEmpty();
        bool isFull();
        int  front();
        int  back();
        void display();

};

#endif
```

Linkedlist.cpp

```
#include<iostream>
#include "linkedlist.h"

using namespace std;


//Checking List is empty or not
bool LinkedList::isEmpty(){
    if(HEAD==nullptr){
        return true;
    }
    else{
        return false;
    }
}


//Adding data to tail of list
void LinkedList::addToTail(int data){
    Node *newNode= new Node(data);
    if(isEmpty()){
        HEAD=newNode;
        TAIL=newNode;
    }
    else{
        TAIL->next=newNode;
        TAIL=newNode;
    }
}



//Removing data from the head of list
void LinkedList::removeFromHead(){
    Node *nodeToDelete =HEAD;
    HEAD=HEAD->next;
    delete nodeToDelete;

}
void LinkedList::traverse(){
   Node* temp;
   temp=HEAD;

   while(temp!=nullptr){
      cout<<temp->data<<endl;
      temp=temp->next;
   }
}
```

Queue_linkedlist.cpp

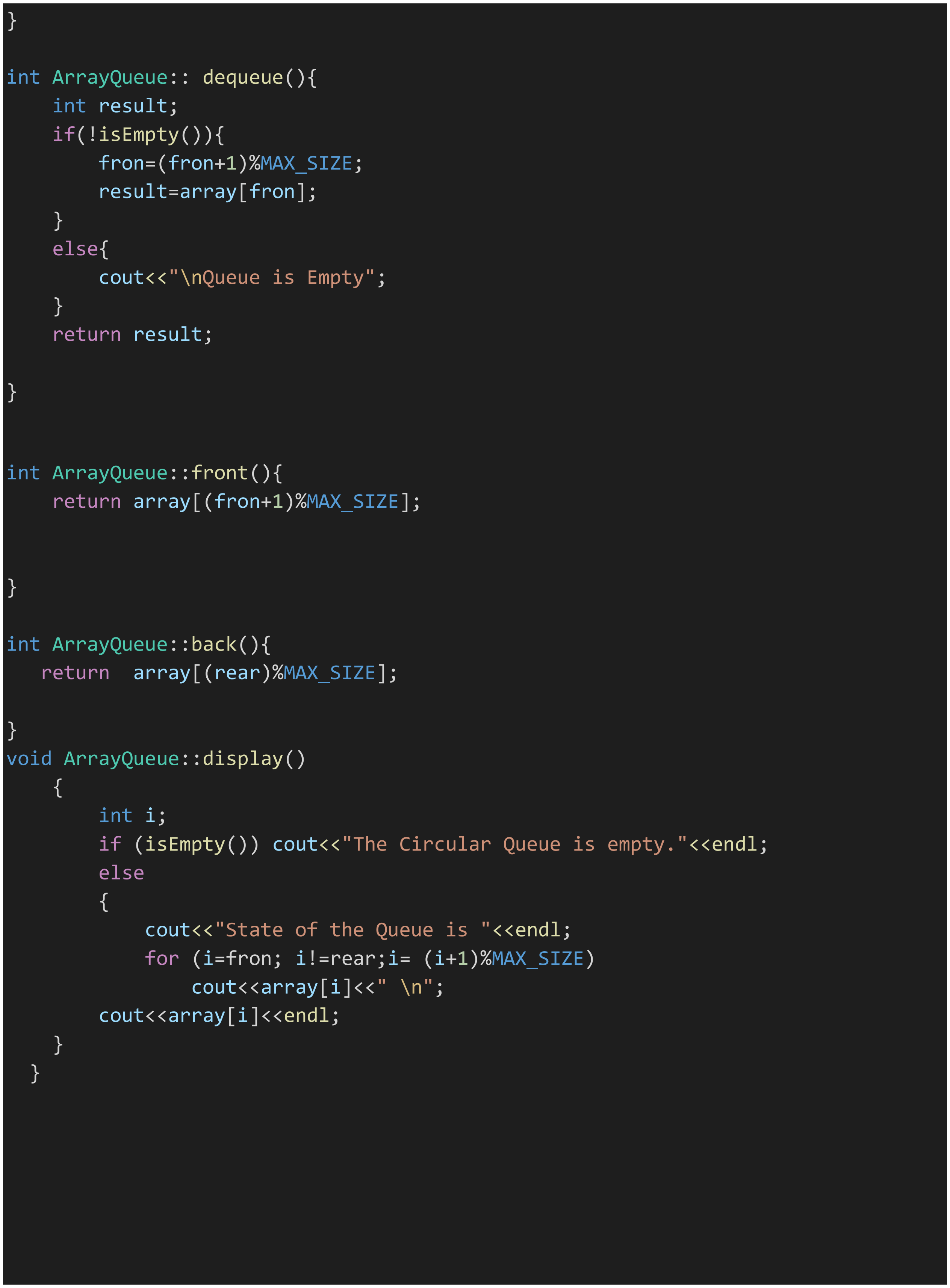
```
#include<iostream>
#include "queue_linkedlist.h"
```

```cpp
#include "linkedlist.h"
using namespace std;


LinkedListQueue::LinkedListQueue()
{

    list.HEAD=NULL;
    list.TAIL=NULL;
}
void LinkedListQueue::enqueue(int data)
{

    if (isFull()){
        cout<<"full"<<endl;
    }
    else{
        list.addToTail(data);
    }


}
int LinkedListQueue::dequeue(){
    int element;
    if(isEmpty()){
        cout<<" empty";
    }
    else
    {
        list.removeFromHead();
    }
    cout<< element;
}
bool LinkedListQueue::isEmpty(){
    list.isEmpty();

}
bool LinkedListQueue::isFull(){
    return 0;
}


int LinkedListQueue::front(){
    return list.HEAD->data;
}
```

```
int LinkedListQueue::back(){
    return list.TAIL->data;
}
void LinkedListQueue::display(){
    list.traverse();
}
```

Queue_array.cpp

```
#include <iostream>
#include "queue_array.h"


using namespace std;


bool ArrayQueue::isEmpty(){
    if(fron==0 && rear==0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool ArrayQueue::isFull(){
    if(rear==MAX_SIZE-1){
        return true;
    }
    else{
        return false;
    }
}

void ArrayQueue:: enqueue(int data){
    if(!isFull()){
        rear=(rear+1)%MAX_SIZE;
        array[rear]=data;
    }
    else{
        cout<<"\nArray is Full"<<endl;
    }
```

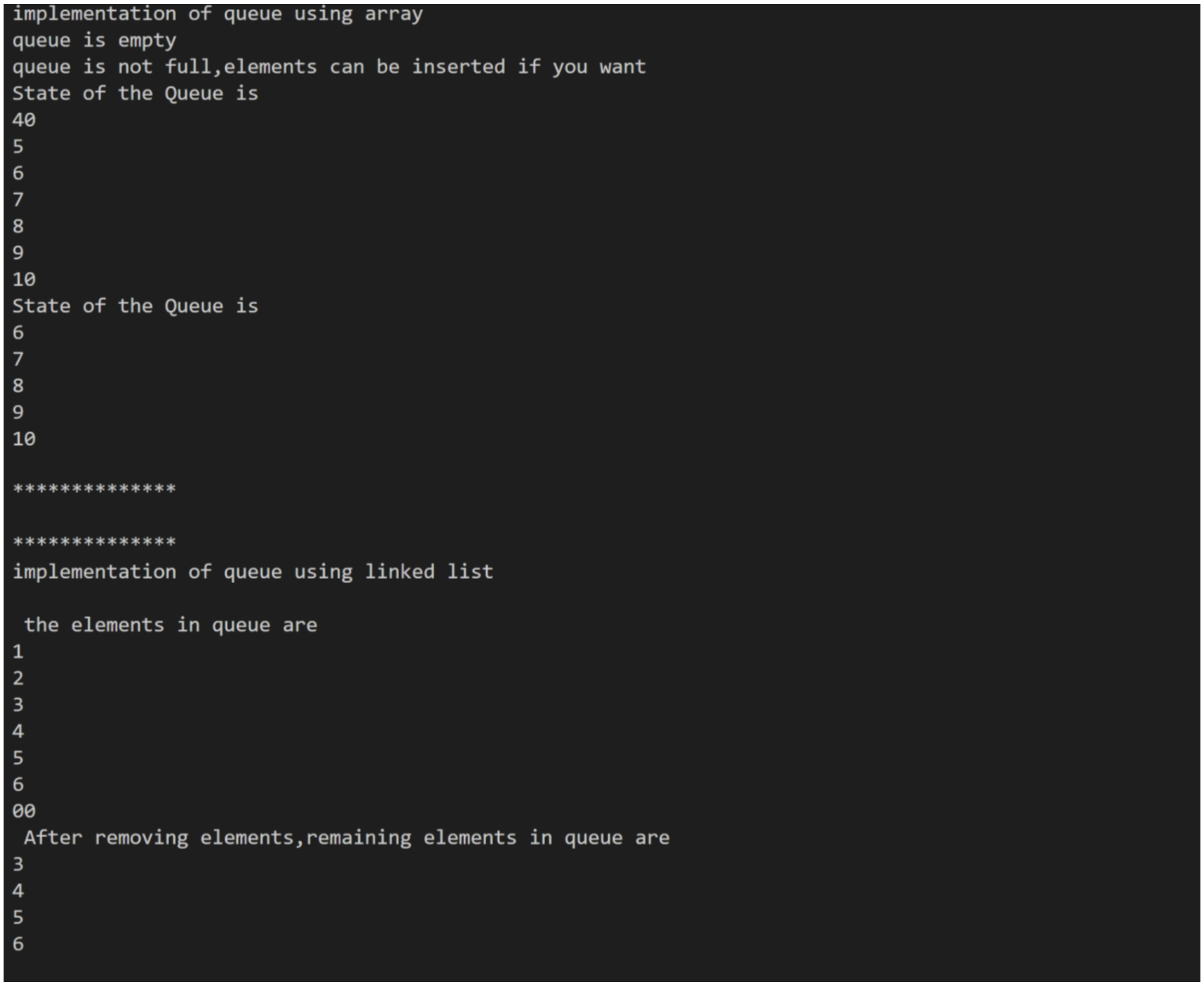


```cpp
}

int ArrayQueue:: dequeue(){
    int result;
    if(!isEmpty()){
        fron=(fron+1)%MAX_SIZE;
        result=array[fron];
    }
    else{
        cout<<"\nQueue is Empty";
    }
    return result;

}


int ArrayQueue::front(){
    return array[(fron+1)%MAX_SIZE];


}

int ArrayQueue::back(){
   return  array[(rear)%MAX_SIZE];

}
void ArrayQueue::display()
    {
        int i;
        if (isEmpty()) cout<<"The Circular Queue is empty."<<endl;
        else
        {
            cout<<"State of the Queue is "<<endl;
            for (i=fron; i!=rear;i= (i+1)%MAX_SIZE)
                cout<<array[i]<<" \n";
        cout<<array[i]<<endl;
    }
  }
```

Main.cpp

```cpp
#include <iostream>
#include "queue.h"

#include "linkedlist.h"
#include "queue_array.h"
#include "queue_linkedlist.h"


using namespace std;

int  main(){
    Queue *queue;
    ArrayQueue array1;
    queue=&array1;
    cout<<"implementation of queue using array"<<endl;
    if(queue->isEmpty()==true){
        cout<<"queue is empty"<<endl;
    }
    else
    cout<<"queue is not empty,elements can be deleted if you want"<<endl;

    if(queue->isFull()==true){
        cout<<"queue is full"<<endl;
    }
    else
     cout<<"queue is not full,elements can be inserted if you want"<<endl;

    queue->enqueue(5);
    queue->enqueue(6);
    queue->enqueue(7);
    queue->enqueue(8);
    queue->enqueue(9);
    queue->enqueue(10);
    queue->display();
    queue->dequeue();
    queue->dequeue();
    queue->display();

    array1.front();
    array1.back();

    cout<<"\n**************"<<endl;
    cout<<"implementation of queue using linked list"<<endl;
    LinkedListQueue linkedlist;
    queue=&linkedlist;
    queue->isEmpty();
    queue->isFull();

    queue->enqueue(1);
    queue->enqueue(2);
    queue->enqueue(3);
    queue->enqueue(4);
    queue->enqueue(5);
    queue->enqueue(6);
    cout<<"\n the elements in queue are"<<endl;
    queue->display();
    queue->dequeue();
    queue->dequeue();
    cout<<"\n After removing elements,remaining elements in queue are"<<endl;
    queue->display();
    cout<<endl;
    queue->front();
    queue->back();

}
```

Output screen



```
implementation of queue using array
queue is empty
queue is not full,elements can be inserted if you want
State of the Queue is
40
5
6
7
8
9
10
State of the Queue is
6
7
8
9
10

**************

**************
implementation of queue using linked list

 the elements in queue are
1
2
3
4
5
6
00
 After removing elements,remaining elements in queue are
3
4
5
6
```