**TITLE OF LAB: (CIRCULAR QUEUES)**

**LAB REPORT NO.09**



**Spring 2022**

**CSE-210L Data Structures and Algorithm Lab**

Submitted by

Name:: **Safi Ullah Khan**

Registration No. **20PWCSE1943**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Khurram Shehzad Khattak**

(Friday, July 29th, 2022)

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**OBJECTIVES OF THE LAB**

‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

In this lab, we will implement and perform basic operations on Circular Queues data structures.

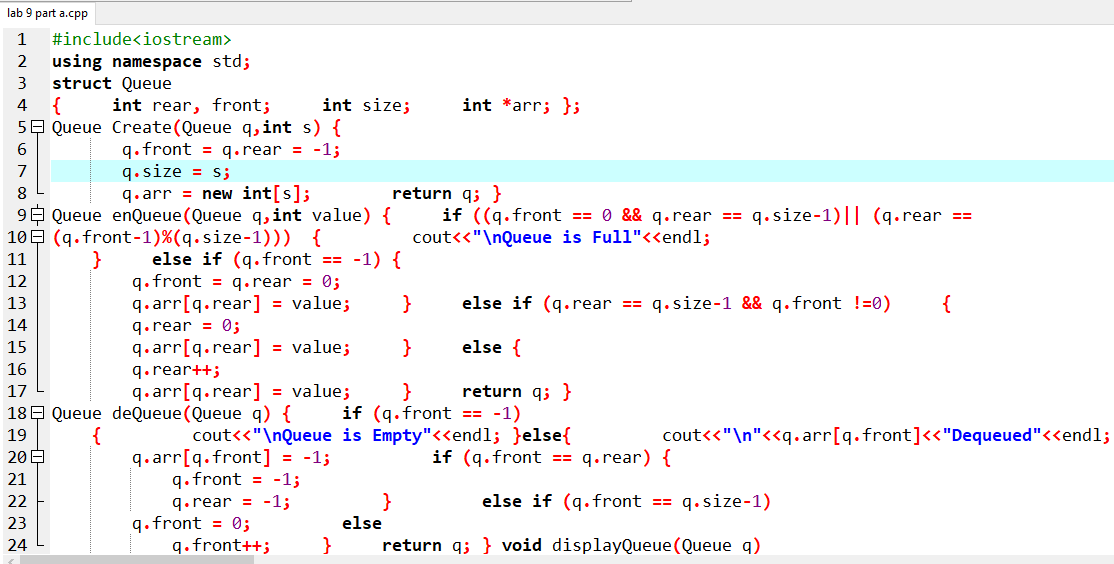
‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

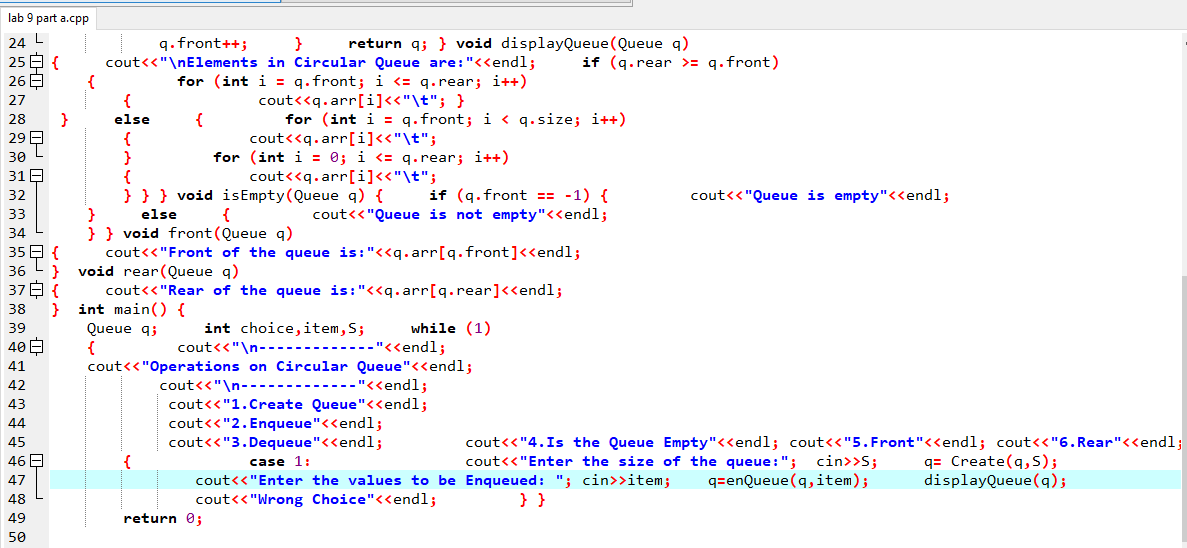
## **Task 01**

Implement following operations of circular queue using arrays

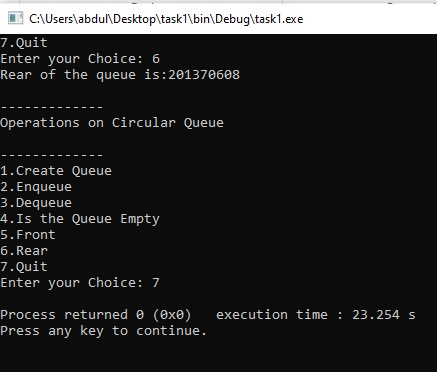
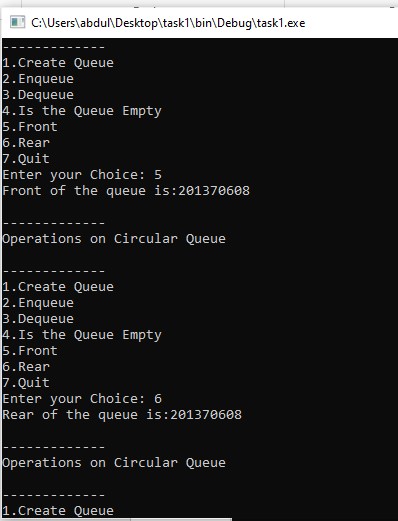
1. Creation
2. Enqueue
3. Dequeue
4. Front
5. Rear
6. Size
7. Empty

**Screenshot of Input:**





**Screenshot of Output:**



## **Task 02**

Implement circular queue with operations from Task1

**Screenshot of Input:**

#include <iostream> using namespace std; typedef struct Node

{ int data;

Node \*next; }Node; class CircularQueue

{ private:

Node \*Front,\*Rear; int Capacity, SIZE; public: void Create(int S) {

Capacity=S;

SIZE=0;

Front=Rear=NULL; }

void Enqueue(int value)

{ if(SIZE==Capacity) cout<<"Queue is Full.\n"; else

{

Node \*temp=new Node; temp->data=value; if(Front==NULL) Front=temp; else

{

Rear->next=temp; Rear=temp;

Rear->next=Front;

} cout<<"Element "<<value<<" enqueued.\n";

SIZE++;

} }

void Dequeue() { if(Front==NULL) cout<<"Queue is Empty.\n"; else { int value; if(Front==Rear)

{ value=Front->data;

free(Front);

Front=Rear=NULL; } else

{

Node \*temp=Front; value=temp->data; Front=Front->next; Rear->next=Front; free(temp);

}

SIZE--; cout<<"Element "<<value<<" Dequeued.\n";

} } void FRONT() { if(Front==NULL) cout<<"Queue is empty.\n"; else

cout<<"Front: "<<Front->data<<endl;

} void REAR() { if(Front==NULL) cout<<"Queue is Empty.\n"; else cout<<"Rear: "<<Rear->data<<endl;

} void Size() { if(Front==NULL) cout<<"Queue is Empty.\n"; else

cout<<"Size of the Queue: "<<SIZE<<endl;

}

void isEmpty()

{ if(Front==NULL) cout<<"Queue is Empty.\n"; else cout<<"Queue is Not Empty.\n";

}

void Display()

{ if(Front==NULL) cout<<"Queue is Empty.\n"; else

{

Node \*temp=Front; while(temp->next!=Front)

{ cout<<temp->data<<"<--"; temp=temp->next;

} cout<<temp->data<<endl; } } }; int main()

{ CircularQueue q; int choice,item,S;

while (1) { cout<<"\n-------------"<<endl; cout<<"Operations on Circular Queue"<<endl; cout<<"\n-------------"<<endl; cout<<"1.Create Queue"<<endl; cout<<"2.Enqueue"<<endl; cout<<"3.Dequeue"<<endl; cout<<"4.Is the Queue Empty"<<endl; cout<<"5.Front"<<endl; cout<<"6.Rear"<<endl; cout<<"7.Display"<<endl; cout<<"8.Quit"<<endl; cout<<"Enter your Choice: "; cin>>choice; switch(choice) { case 1:

cout<<"Enter the size of the queue: "; cin>>S;

q.Create(S); break; case 2:

cout<<"Enter the values to be Enqueued: "; cin>>item;

q.Enqueue(item); break; case 3:

q.Dequeue(); break; case 4:

q.isEmpty(); break; case5:

q.FRONT(); break;

case 6:

q.REAR(); break; case 7:

q.Display(); break; case 8: return 0; break; default: cout<<"Wrong Choice"<<endl;

}

} return 0;

}

**Screenshot of Output:**

