Government College of Engineering, Karad Programming for Problem Solving Lab

Nanekar Saurabh Rajesh

20141212

I1

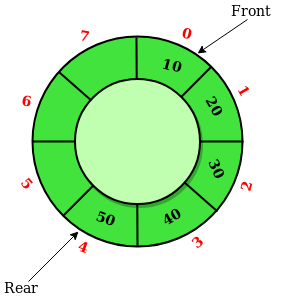
**Experiment No .2**

**Title**: Implement circular queue as an ADT using array.

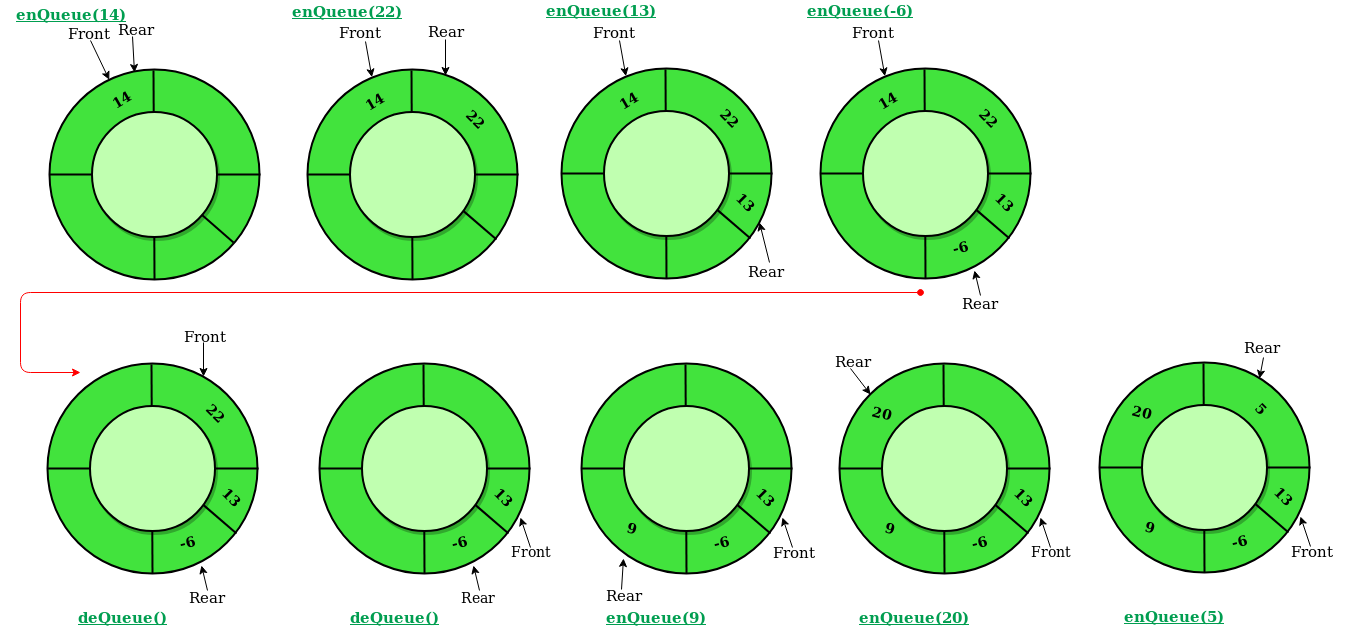
**Outcome:** Students can demonstrate circular queue operations using array.

**Theory:**

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **‘Ring Buffer’**. 



In a normal Queue, we can insert elements until queue becomes full. But once queue becomes full, we cannot insert the next element even if there is a space in front of queue.



**Operations on Circular Queue:**

* Front: Get the front item from queue.
* Rear: Get the last item from queue.
* **enQueue(value) This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at Rear position.**

Check whether queue is Full – Check ((rear == SIZE-1 && front == 0) || (rear == front-1)).

If it is full then display Queue is full. If queue is not full then, check if (rear == SIZE – 1 && front != 0)

If it is true then set rear=0 and insert element.

* **deQueue() This function is used to delete an element from the circular queue.**

In a circular queue, the element is always deleted from front position.

Check whether queue is Empty means check (front==-1).

If it is empty then display Queue is empty. If queue is not empty then step 3

Check if (front==rear) if it is true then set front=rear= -1 else check if (front==size-1), if it is true then set front=0 and return the element.

* **display() -Displays the elements of a Circular Queue**

Step 1: Check whether queue is EMPTY. (front == -1)

Step 2: If it is EMPTY, then display "Queue is EMPTY!!!" and terminate the function.

Step 3: If it is NOT EMPTY, then define an integer variable 'i' and set 'i = front'.

Step 4: Check whether 'front <= rear', if it is TRUE, then display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until 'i <= rear' becomes FALSE.

Step 5: If 'front <= rear' is FALSE, then display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until'i <= SIZE - 1' becomes FALSE.

Step 6: Set i to 0.

Step 7: Again display 'cQueue[i]' value and increment i value by one (i++). Repeat the same until 'i <= rear' becomes FALSE.

**Applications of a Circular Queue**

* Memory management: circular queue is used in memory management.
* Process Scheduling: A CPU uses a queue to schedule processes.
* Traffic Systems: Queues are also used in traffic systems.

**Analysis:**



**List of similar programs: Solve any one.**

1. Write a program for implementation of stack using queue.
2. Write a program for implementation of priority queue.
3. Write a program for implementation of queue using stack.

**Title Program: Implement circular queue as an ADT using array.**

**Source code of Implemented Programs:**

Code for Circular Queue:-

//Nanekar Saurabh Rajesh

#include<stdio.h>

#include<stdlib.h>

int cqueue[50],i,choice,n;

int front=-1,rear=-1;

void insert(void);

void delete(void);

void display(void);

int main()

{

printf("Enter the size of cqueue:-");

scanf("%d",&n);

do

{

printf("\nChoose choice from below:-");

printf("\n1.Insert\n2.Delete\n3.Display");

printf("\nEnter your choice:-");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

{

insert();

break;

}

case 2:

{

delete();

break;

}

default:

{

printf("Choose valid choice from 1,2");

}

}

}

while (choice!=2);

return 0;

}

void insert()

{

int item;

printf("\nEnter the element:-");

scanf("%d",&item);

if(rear == n-1&&front==0 ||front==rear+1)

{

printf("\nOverflow");

}

if(rear==n-1 && front!=0)

{

rear=-1;

}

else

{

rear=rear+1;

}

cqueue[rear]=item;

printf("\n Value inserted.");

}

void delete()

{

int item;

if(front==-1 && rear==-1)

{

printf("\nUnderflow");

}

else

{

item=cqueue[front];

cqueue[front]=-1;

if(front==n-1)

{

front=0;

}

else if(front==rear)

{

front=-1;

rear=-1;

}

else

{

front++;

}

printf("\nValue deleted from queue is%d",cqueue[front]);

}

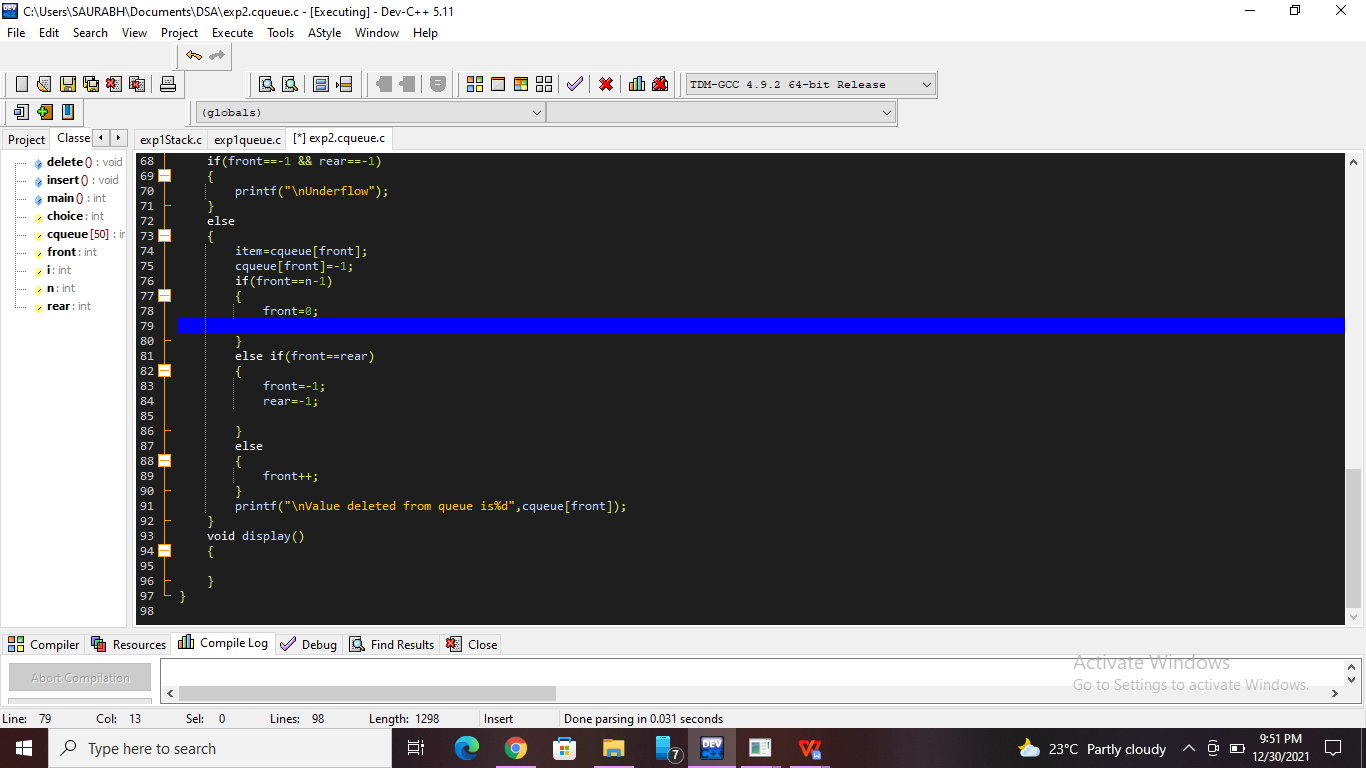
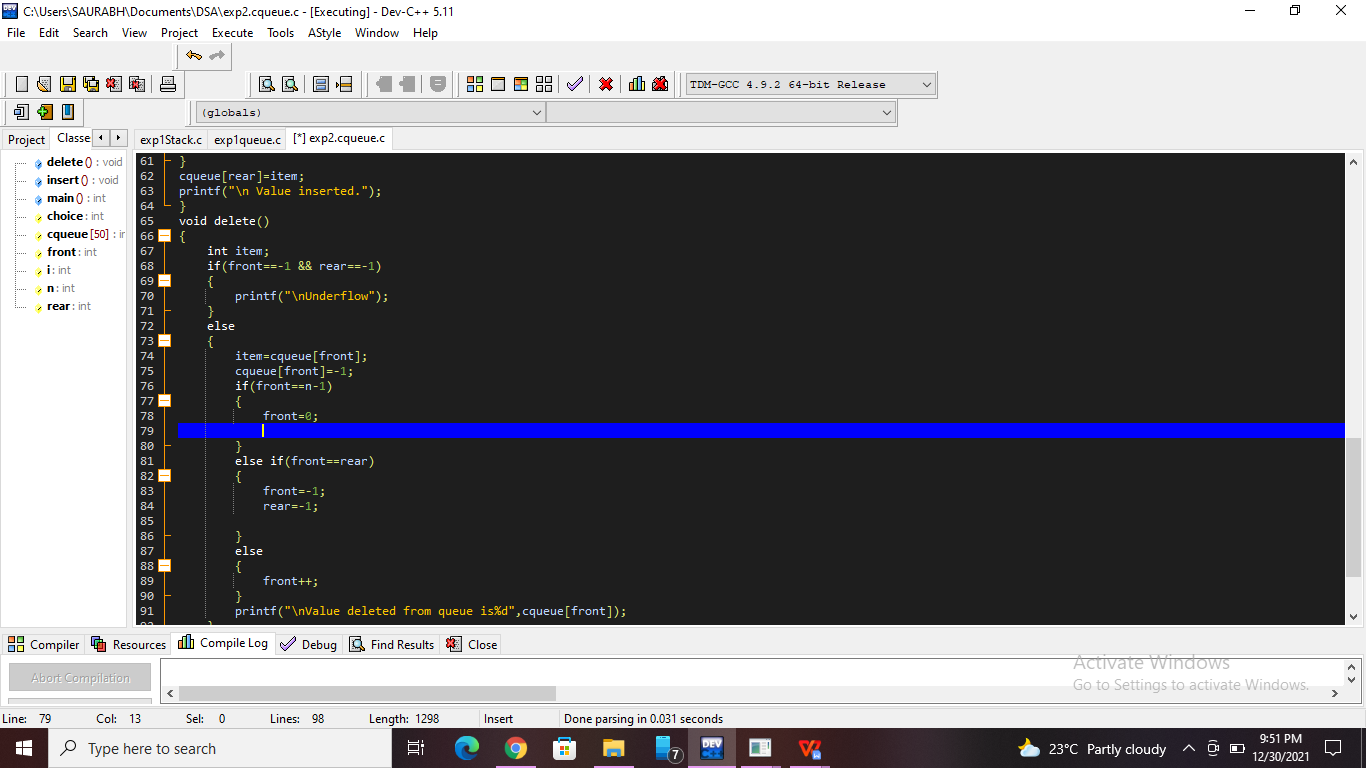
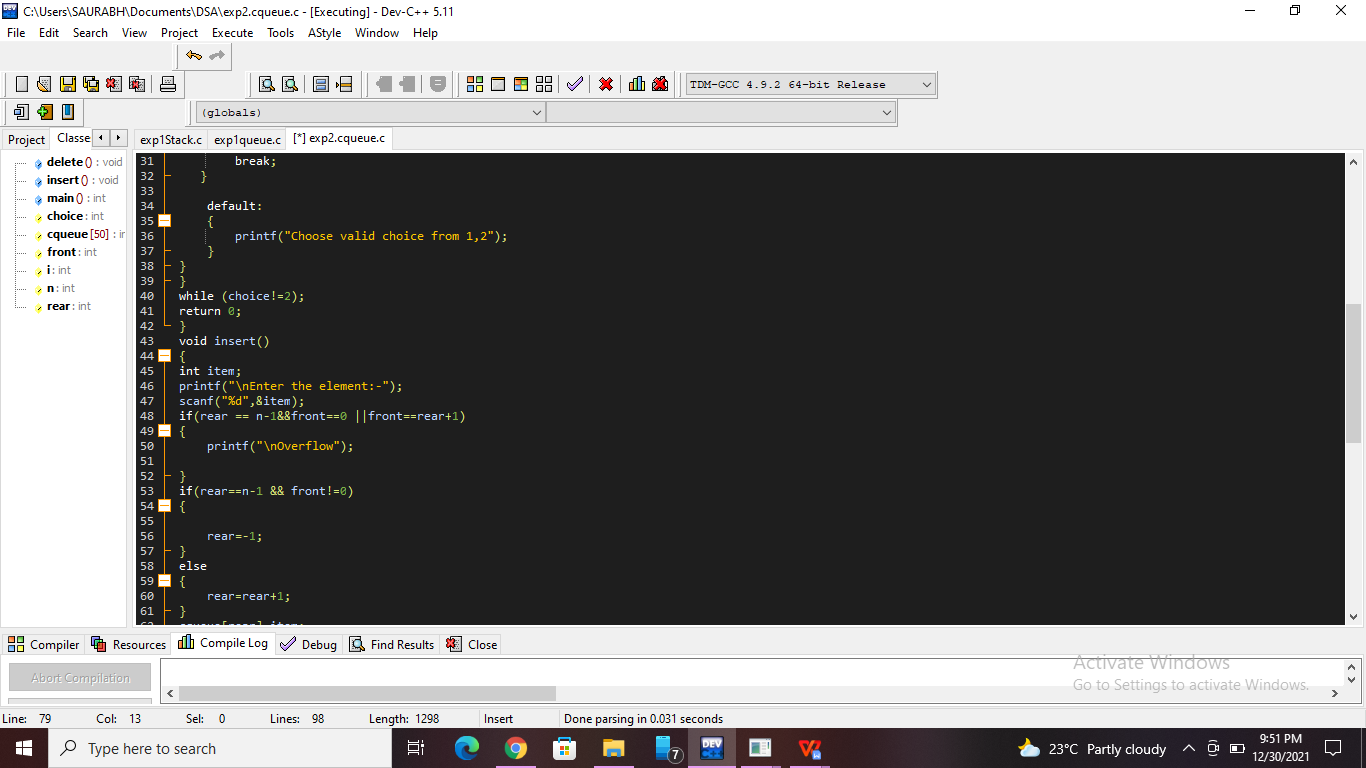
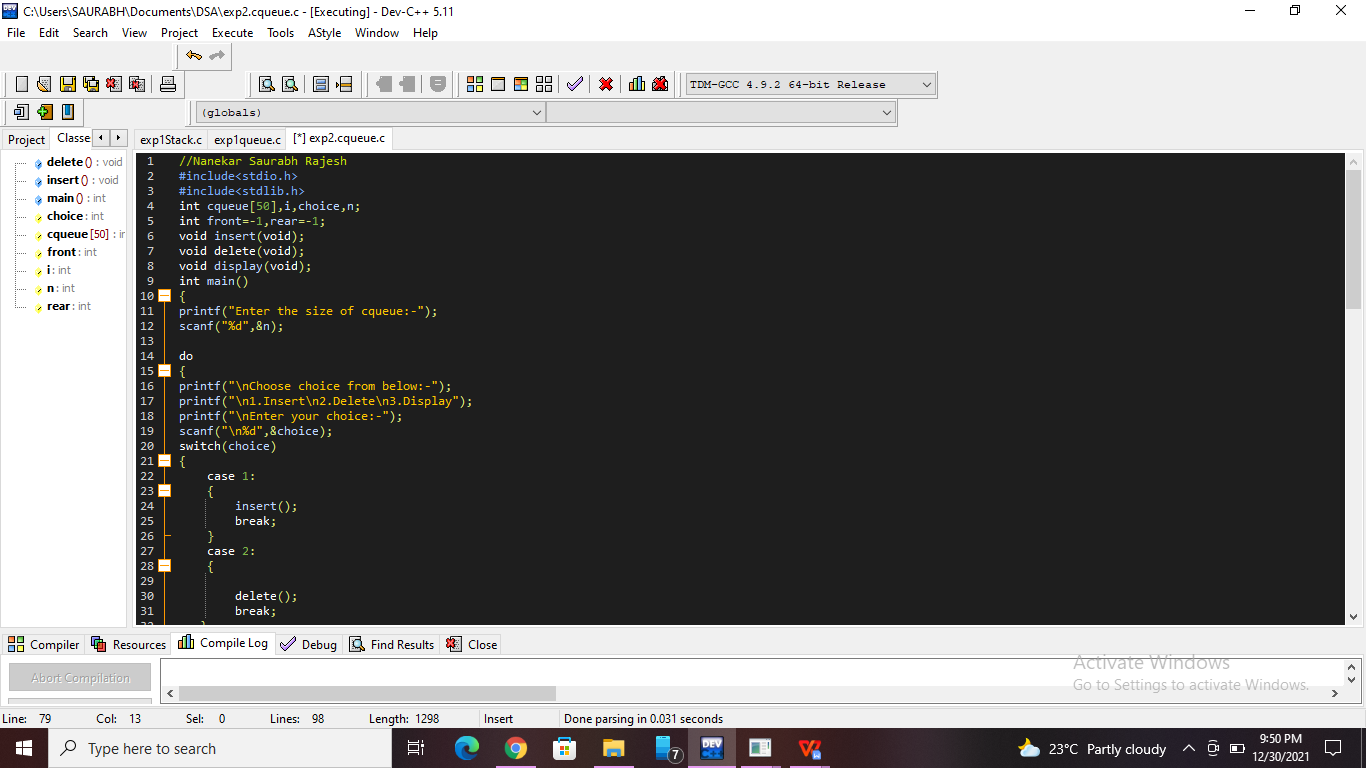
void display()

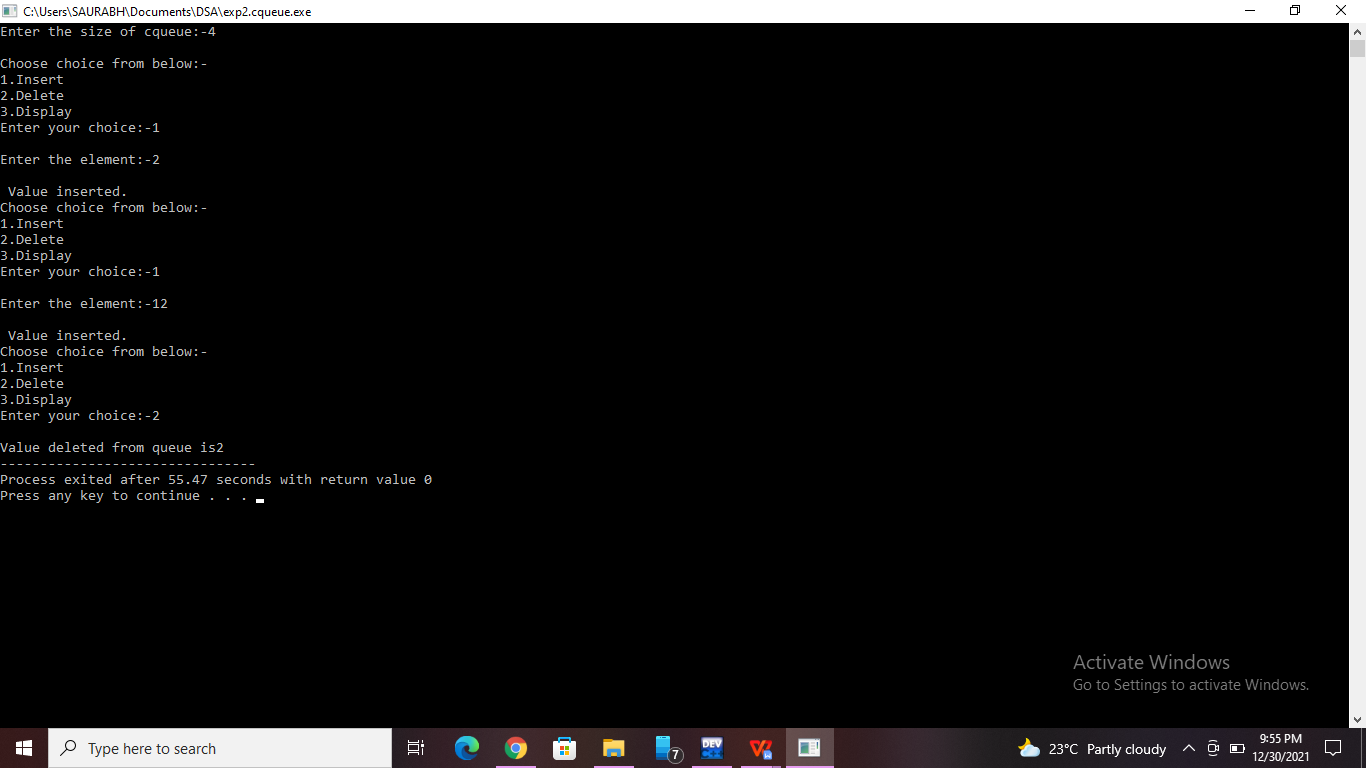
{

}

}

**Screenshots of Input/Output:**





**Practice Program:**

Write a program for implementation of priority queue.

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

void insert\_by\_priority(int);

void delete\_by\_priority(int);

void create();

void check(int);

void display\_pqueue();

int pri\_que[MAX];

int front, rear;

void main()

{

    int n, ch;

    printf("\n1 - Insert an element into queue");

    printf("\n2 - Delete an element from queue");

    printf("\n3 - Display queue elements");

    printf("\n4 - Exit");

    create();

    while (1)

    {

        printf("\nEnter your choice : ");

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

            printf("\nEnter value to be inserted : ");

            scanf("%d",&n);

            insert\_by\_priority(n);

            break;

        case 2:

            printf("\nEnter value to delete : ");

            scanf("%d",&n);

            delete\_by\_priority(n);

            break;

        case 3:

            display\_pqueue();

            break;

        case 4:

            exit(0);

        default:

            printf("\nChoice is incorrect, Enter a correct choice");

        }

    }

}

void create()

{

    front = rear = -1;

}

void insert\_by\_priority(int data)

{

    if (rear >= MAX - 1)

    {

        printf("\nQueue overflow no more elements can be inserted");

        return;

    }

    if ((front == -1) && (rear == -1))

    {

        front++;

        rear++;

        pri\_que[rear] = data;

        return;

    }

    else

        check(data);

    rear++;

}

void check(int data)

{

    int i,j;

    for (i = 0; i <= rear; i++)

    {

        if (data >= pri\_que[i])

        {

            for (j = rear + 1; j > i; j--)

            {

                pri\_que[j] = pri\_que[j - 1];

            }

            pri\_que[i] = data;

            return;

        }

    }

    pri\_que[i] = data;

}

void delete\_by\_priority(int data)

{

    int i;

    if ((front==-1) && (rear==-1))

    {

        printf("\nQueue is empty no elements to delete");

        return;

    }

    for (i = 0; i <= rear; i++)

    {

        if (data == pri\_que[i])

        {

            for (; i < rear; i++)

            {

                pri\_que[i] = pri\_que[i + 1];

            }

        pri\_que[i] = -99;

        rear--;

        if (rear == -1)

            front = -1;

        return;

        }

    }

    printf("\n%d not found in queue to delete", data);

}

void display\_pqueue()

{

    if ((front == -1) && (rear == -1))

    {

        printf("\nQueue is empty");

        return;

    }

    for (; front <= rear; front++)

    {

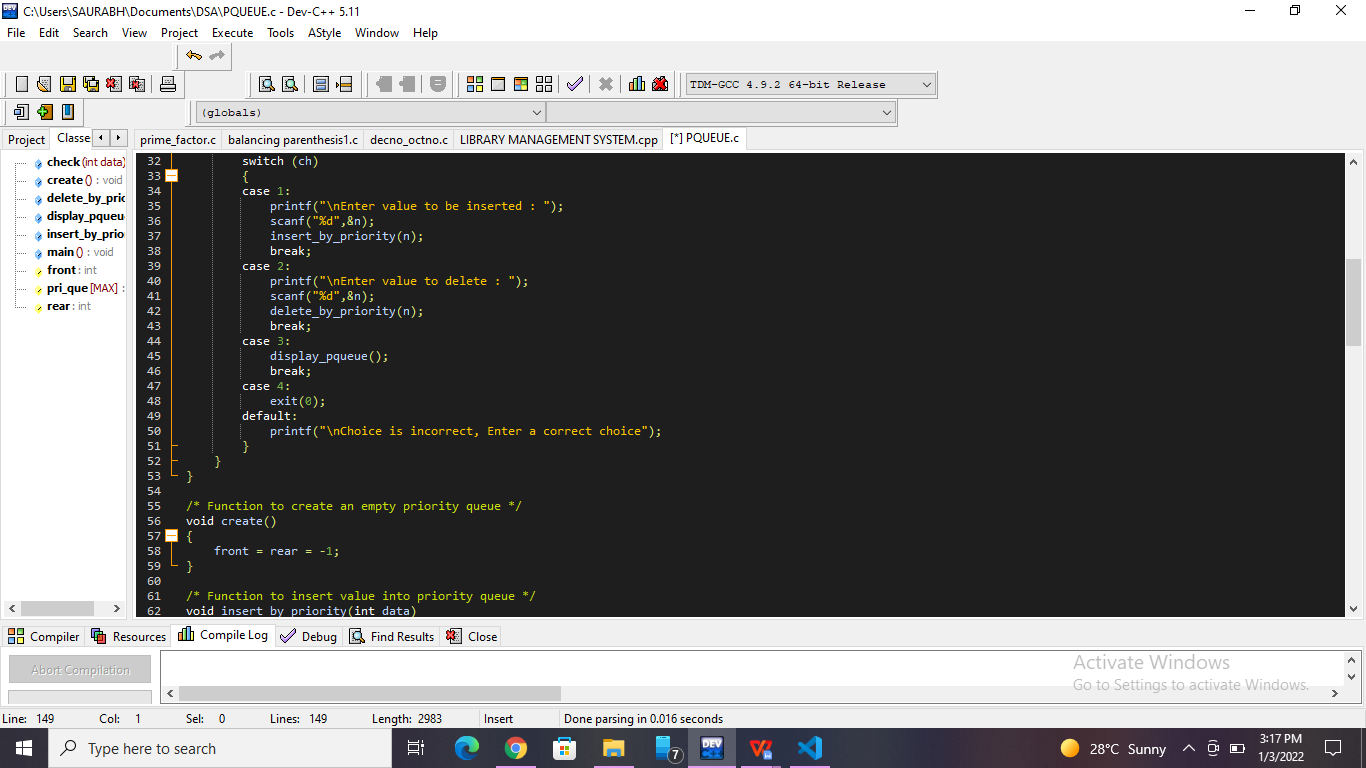
        printf(" %d ", pri\_que[front]);

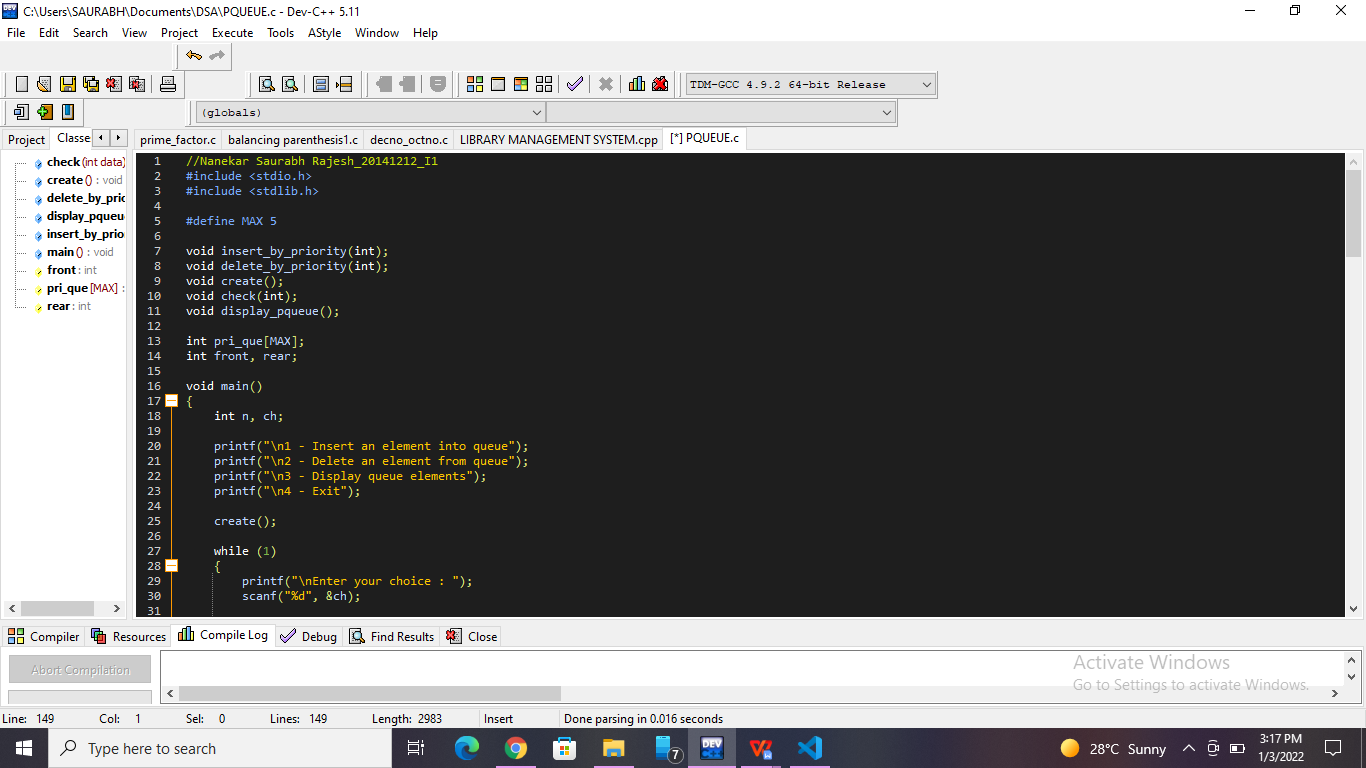
    }

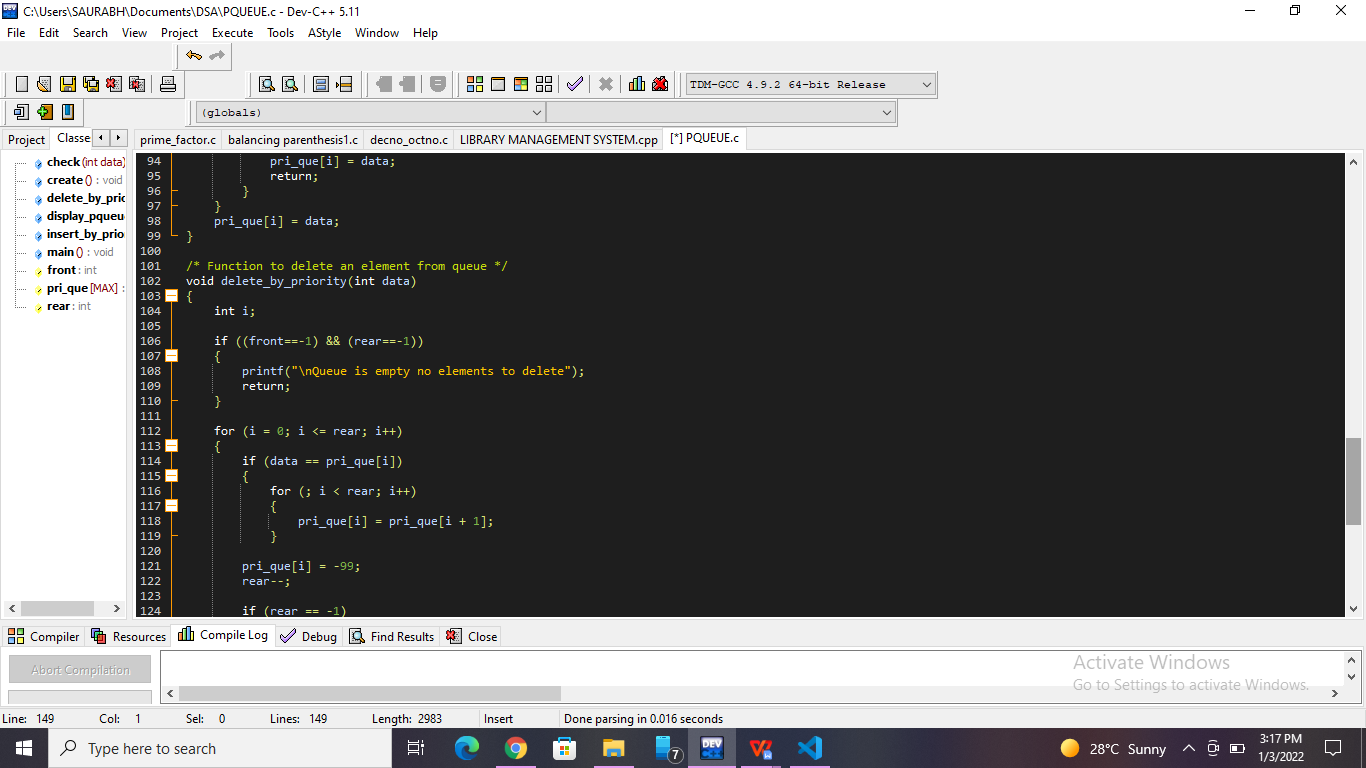
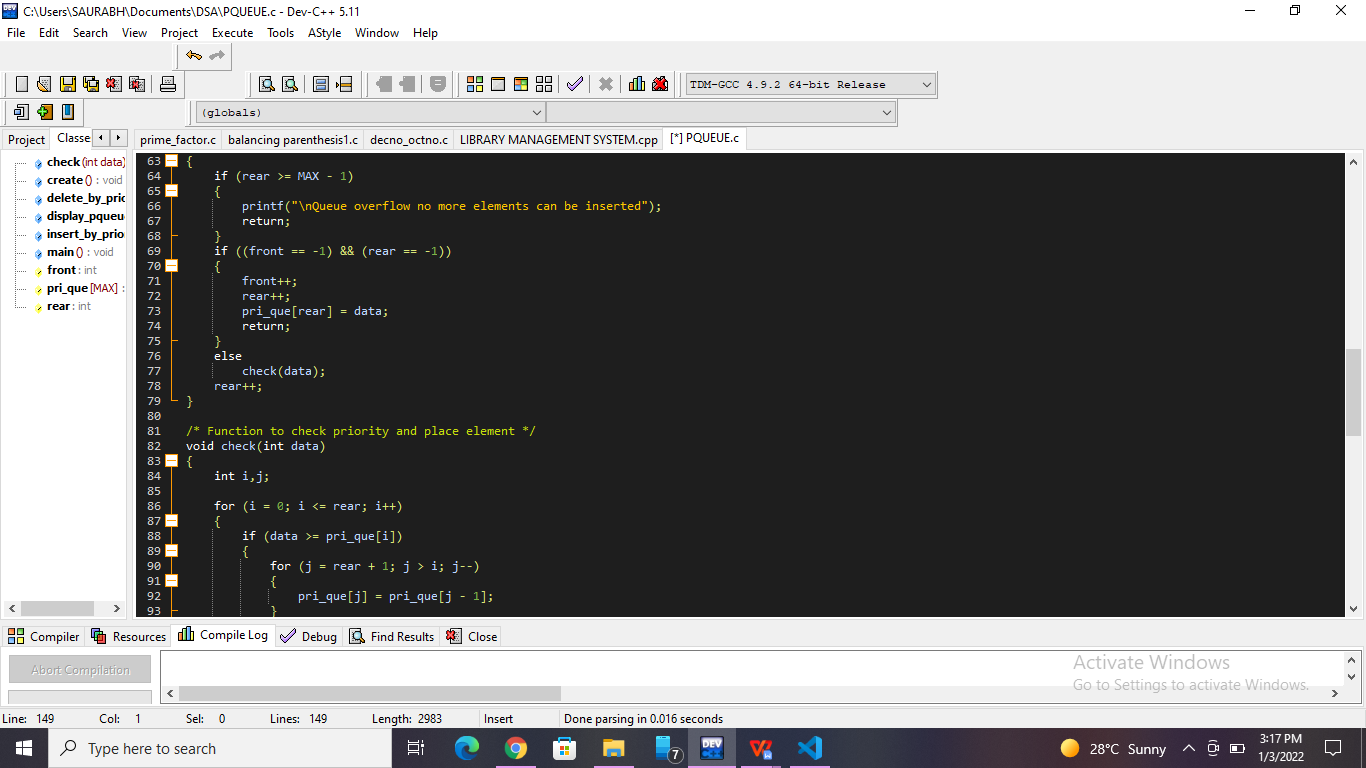
    front = 0;

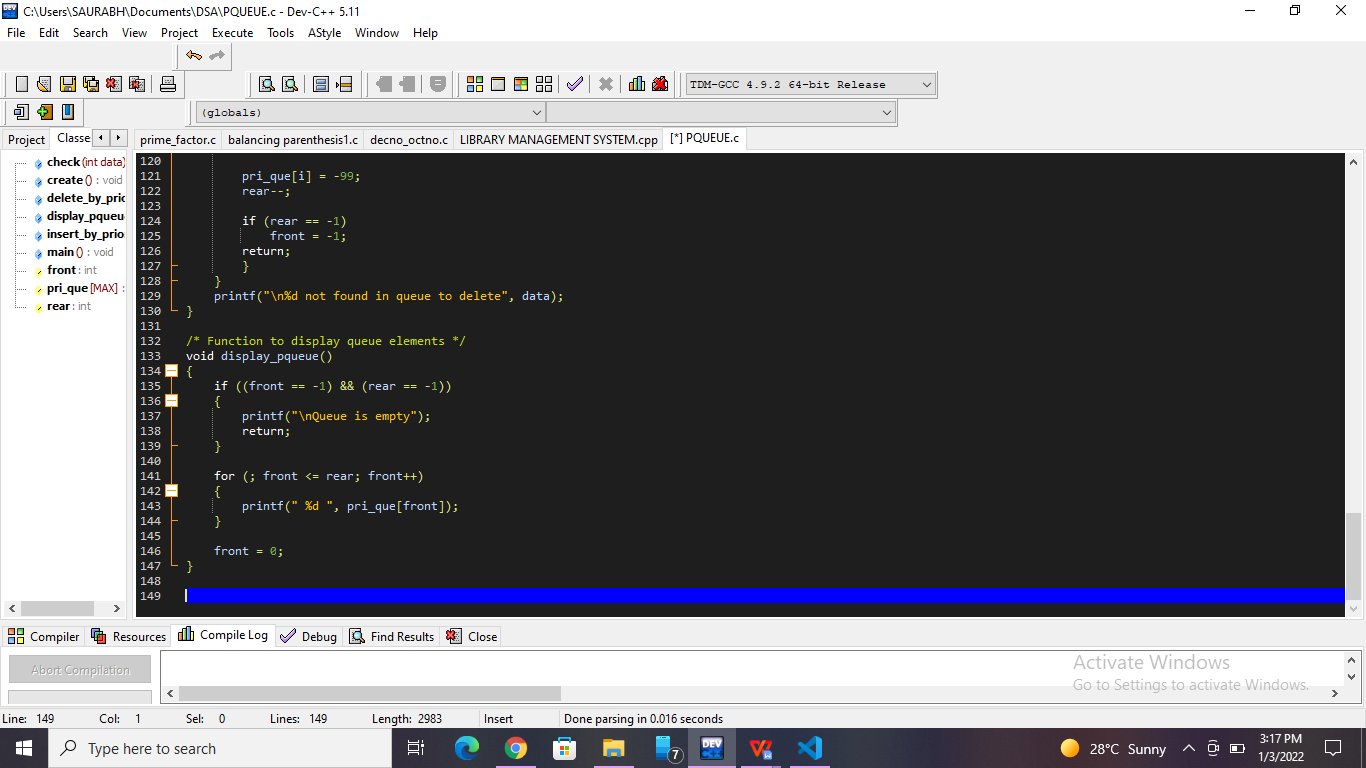
}

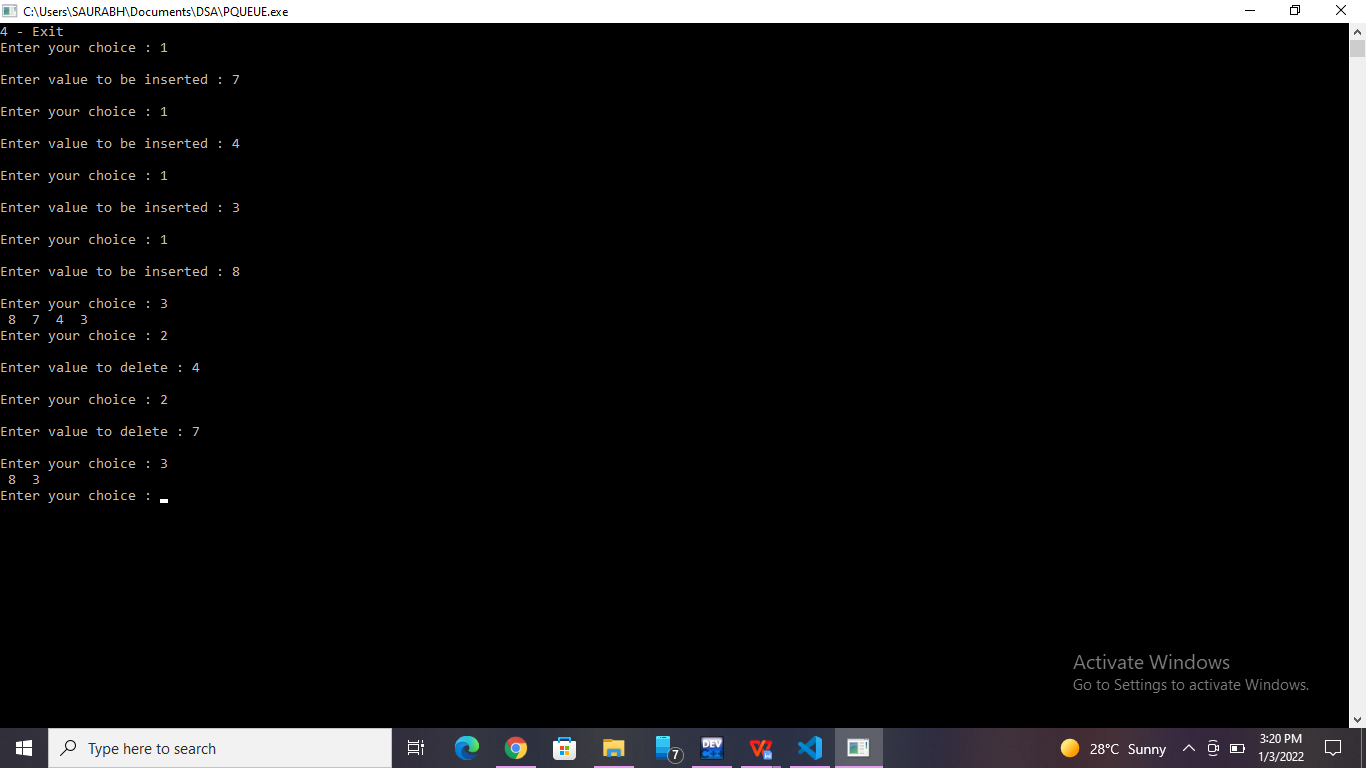
**Screenshots of Practice Program:-**

****

****

****

****

****

**List of sample questions for oral examination:**

1. What can we say about the array representation of a circular queue when it contains only one element?
2. What is the need for circular array to implement queue?
3. What is queue full condition if it is implemented with an array?
4. What are the applications of circular queue?
5. How a circular queue can be implemented using array?

**Conclusion:**

Circular queue is better than a normal queue because in the former we can effectively utilize the memory space. If we have a normal queue and have deleted some elements from there then empty space is created there and even if the queue has empty cells then also we cannot insert any new element because the insertion has to be done from one side only(i.e rear or tail) and deletion has to be done from another side(i.e front or head). But in case of circular queue the front and rear are adjacent to each other.