Green University of Bangladesh

Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Summer , Year:2022), B.Sc. in CSE (Day)

LAB REPORT NO : 4

Course Title: Structured Programming Lab

Course Code: CSE 106 Section:PC-213DA

Lab Experiment Name: Infix to Postfix & Circular Queue

<u>Student Details</u>

Name	ID
MD Dulal Hossain	213902116

Lab Date : 27-07-2022
Submission Date : 07-08-2022
Course Teacher's Name : Farhana Akther Sunny

[For Teachers use only: Don't Write Anything inside this box]

<u>Lab Report Status</u>

Marks: ………………………………….	**Signature:.....................**
Comments:...	**Date:..............................**

//Infix to Postfix

Algorithm :

Step 1 : Scan the Infix Expression from left to right.
Step 2 : If the scanned character is an operand, append it with final Infix to Postfix string.
Step 3 : Else,
Step 3.1 : If the precedence order of the scanned(incoming) operator is greater than the precedence order of the operator in the stack (or the stack is empty or the stack contains a ‘(‘ push it on stack.
Step 3.2 : Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)
Step 4 : If the scanned character is an ‘(‘ push it to the stack.
Step 5 : If the scanned character is an ‘)’ pop the stack and and output it until a ‘(‘ respectively is encountered, and discard both the parenthesis.
Step 6 : Repeat steps 2-6 until infix expression is scanned.
Step 7 : Print the output
Step 8 : Pop and output from the stack until it is not empty.

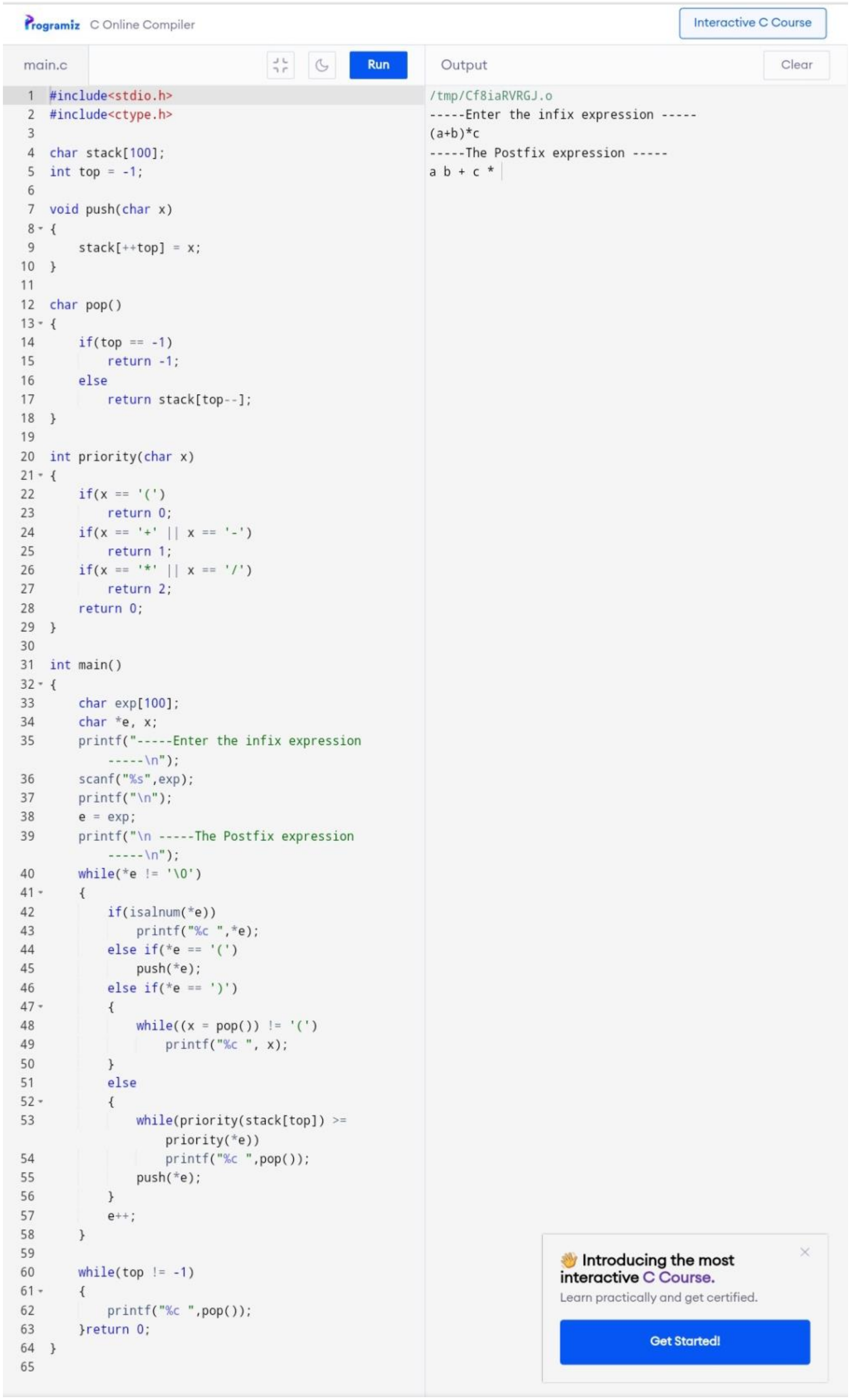
Source Code :

```
#include<stdio.h>
#include<ctype.h>

Char stack[100];
Int top = -1;
```

```
Void push(char x)
{
   Stack[++top] = x;
}

Char pop()
{
   If(top == -1)
      Return -1;
   Else
      Return stack[top--];
}

Int priority(char x)
{
   If(x == ‘(‘)
      Return 0;
   If(x == ‘+’ || x == ‘-‘)
      Return 1;
   If(x == ‘*’ || x == ‘/’)
      Return 2;
   Return 0;
}
Int main()
{
   Char exp[100];
   Char *e, x;
   Printf(“-----Enter the infix expression -----\n”);
   Scanf(“%s”,exp);
   Printf(“\n”);
   E = exp;
   Printf(“\n -----The Postfix expression -----\n”);
```

```
    While(*e != ‘\0’)
    {
        If(isalnum(*e))
            Printf(“%c “,*e);
        Else if(*e == ‘(‘)
            Push(*e);
        Else if(*e == ‘)’)
        {
            While((x = pop()) != ‘(‘)
                Printf(“%c “, x);
        }
        Else
        {
            While(priority(stack[top]) >= priority(*e))
                Printf(“%c “,pop());
            Push(*e);
        }
        E++;
    }

    While(top != -1)
    {
        Printf(“%c “,pop());
    }return 0;
}
```

My Code & Output



```c
#include<stdio.h>
#include<ctype.h>

char stack[100];
int top = -1;

void push(char x)
{
    stack[++top] = x;
}

char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}

int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}

int main()
{
    char exp[100];
    char *e, x;
    printf("-----Enter the infix expression
        -----\n");
    scanf("%s",exp);
    printf("\n");
    e = exp;
    printf("\n -----The Postfix expression
        -----\n");
    while(*e != '\0')
    {
        if(isalnum(*e))
            printf("%c ",*e);
        else if(*e == '(')
            push(*e);
        else if(*e == ')')
        {
            while((x = pop()) != '(')
                printf("%c ", x);
        }
        else
        {
            while(priority(stack[top]) >=
                priority(*e))
                printf("%c ",pop());
            push(*e);
        }
        e++;
    }

    while(top != -1)
    {
        printf("%c ",pop());
    }return 0;
}

```

```
/tmp/Cf8iaRVRGJ.o
-----Enter the infix expression -----
(a+b)*c
-----The Postfix expression -----
a b + c *
```

👋 Introducing the most interactive C Course.
Learn practically and get certified.
Get Started!

//Circular Queue

Algorithm :

Insert

```
Step 1: if (rear+1)%max = front
write " overflow "
goto step 4
[end of if]

Step 2: if front = -1 and rear = -1
set front = rear = 0
else if rear = max – 1 and front ! = 0
set rear = 0
else
set rear = (rear + 1) % max
[end of if]

Step 3: set queue[rear] = val
Step 4: Exit
```

Delete

```
Step 1: if front = -1
write " underflow "
goto step 4
[end of if]

Step 2: set val = queue[front]
Step 3: if front = rear
set front = rear = -1
```

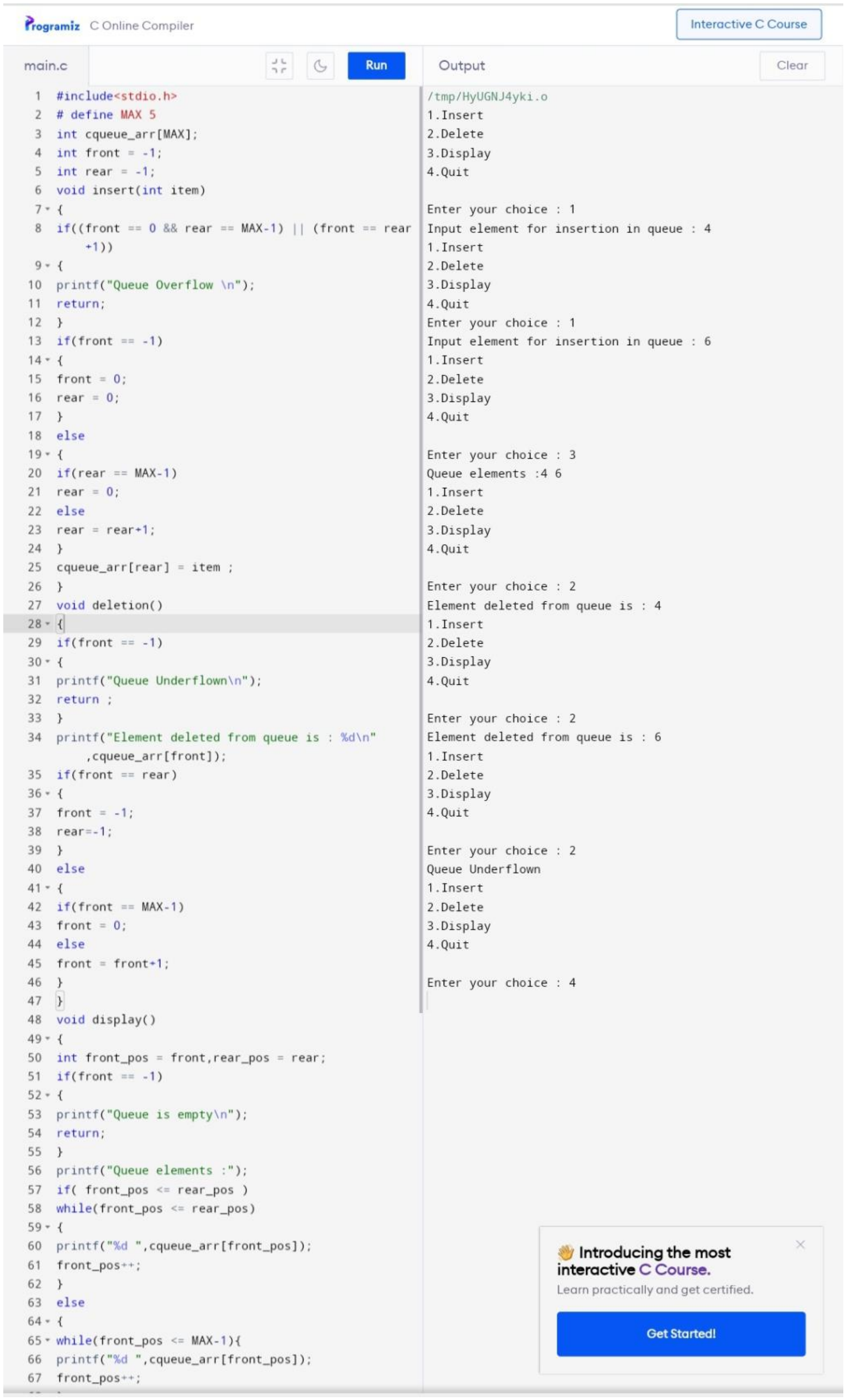
```
else
if front = max -1
set front = 0
else
set front = front + 1
[end of if]
[end of if]

Step 4: exit
```

Source Code :

```
#include<stdio.h>
# define MAX 5
Int cqueue_arr[MAX];
Int front = -1;
Int rear = -1;
Void insert(int item)
{
If((front == 0 && rear == MAX-1) || (front == rear+1))
{
Printf("Queue Overflow \n");
Return;
}
If(front == -1)
{
Front = 0;
Rear = 0;
}
Else
{
If(rear == MAX-1)
Rear = 0;
Else
Rear = rear+1;
}
Cqueue_arr[rear] = item ;
}
Void deletion()
{
If(front == -1)
{
Printf("Queue Underflown\n");
Return ;
}
Printf("Element deleted from queue is : %d\n",cqueue_arr[front]);
If(front == rear)
{
Front = -1;
Rear=-1;
}
Else
{
If(front == MAX-1)
Front = 0;
Else
Front = front+1;
}
}
Void display()
{
Int front_pos = front,rear_pos = rear;
If(front == -1)
{
Printf("Queue is empty\n");
Return;
}
Printf("Queue elements :");
If( front_pos <= rear_pos )
While(front_pos <= rear_pos)
```

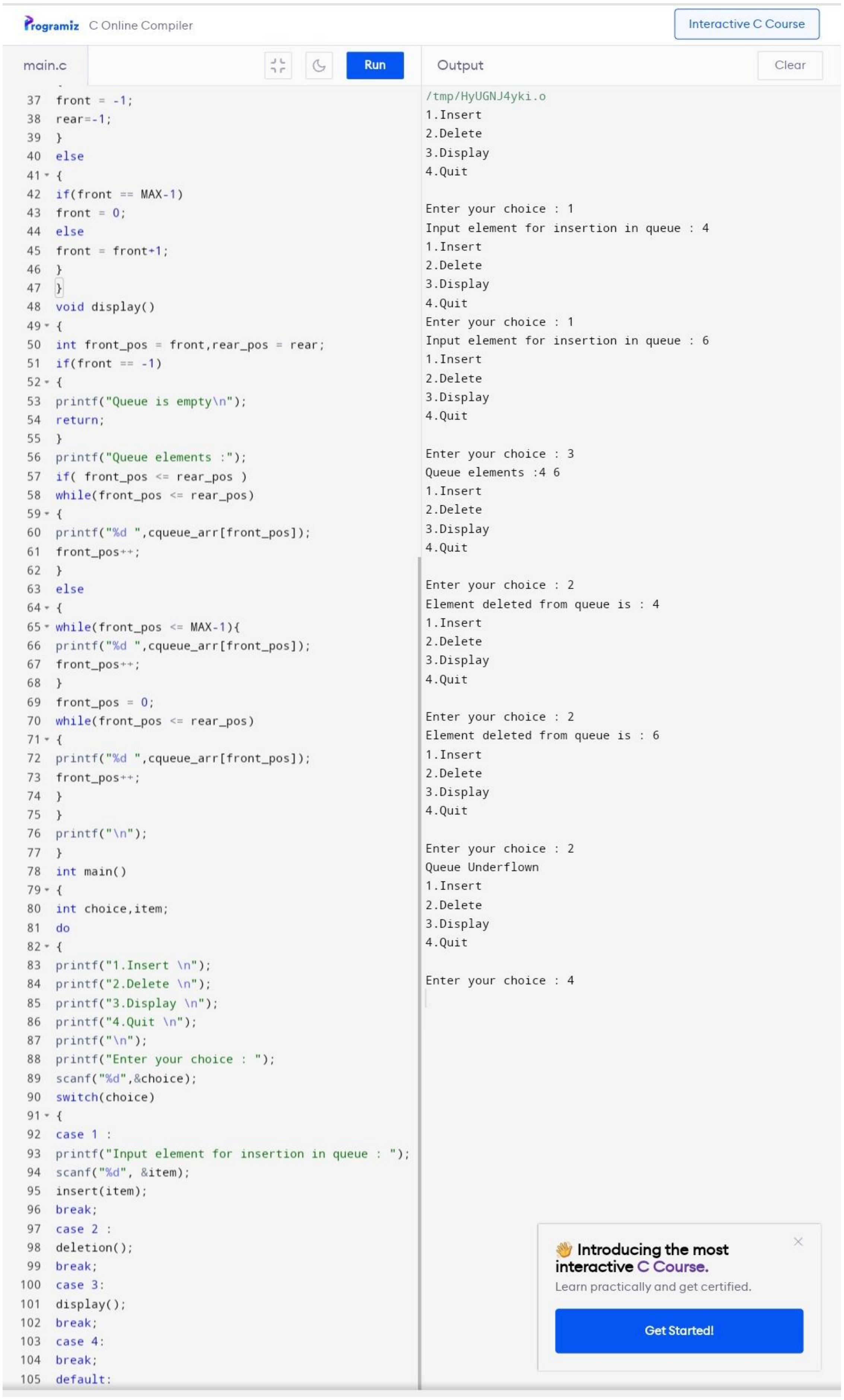
```
{
Printf("%d ",cqueue_arr[front_pos]);
Front_pos++;
}
Else
{
While(front_pos <= MAX-1){
Printf("%d ",cqueue_arr[front_pos]);
Front_pos++;
}
Front_pos = 0;
While(front_pos <= rear_pos)
{
Printf("%d ",cqueue_arr[front_pos]);
Front_pos++;
}
}
Printf("\n");
}
Int main()
{
Int choice,item;
Do
{
Printf("1.Insert \n");
Printf("2.Delete \n");
Printf("3.Display \n");
Printf("4.Quit \n");
Printf("\n");
Printf("Enter your choice : ");
Scanf("%d",&choice);
Switch(choice)
{
Case 1 :
Printf("Input element for insertion in queue : ");
Scanf("%d", &item);
Insert(item);
Break;
Case 2 :
Deletion();
Break;
Case 3:
Display();
Break;
Case 4:
Break;
Default:
Printf("Wrong choice\n");
}
}while(choice!=4);
Return 0;
}
```

My code & Output


Programiz C Online Compiler
Interactive C Course
main.c
Run
Output
Clear
Introducing the most interactive C Course.
Learn practically and get certified.
Get Started!

```c
#include<stdio.h>
# define MAX 5
int cqueue_arr[MAX];
int front = -1;
int rear = -1;
void insert(int item)
{
if((front == 0 && rear == MAX-1) || (front == rear
    +1))
{
printf("Queue Overflow \n");
return;
}
if(front == -1)
{
front = 0;
rear = 0;
}
else
{
if(rear == MAX-1)
rear = 0;
else
rear = rear+1;
}
cqueue_arr[rear] = item ;
}
void deletion()
{
if(front == -1)
{
printf("Queue Underflown\n");
return ;
}
printf("Element deleted from queue is : %d\n"
    ,cqueue_arr[front]);
if(front == rear)
{
front = -1;
rear=-1;
}
else
{
if(front == MAX-1)
front = 0;
else
front = front+1;
}
}
void display()
{
int front_pos = front,rear_pos = rear;
if(front == -1)
{
printf("Queue is empty\n");
return;
}
printf("Queue elements :");
if( front_pos <= rear_pos )
while(front_pos <= rear_pos)
{
printf("%d ",cqueue_arr[front_pos]);
front_pos++;
}
else
{
while(front_pos <= MAX-1){
printf("%d ",cqueue_arr[front_pos]);
front_pos++;
```

```
/tmp/HyUGNJ4yki.o
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 1
Input element for insertion in queue : 4
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input element for insertion in queue : 6
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 3
Queue elements :4 6
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 2
Element deleted from queue is : 4
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 2
Element deleted from queue is : 6
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 2
Queue Underflown
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 4
```


Programiz
C Online Compiler
Interactive C Course
main.c
Run
Output
Clear
👋 Introducing the most interactive C Course.
Learn practically and get certified.
Get Started!

```c
front = -1;
rear=-1;
}
else
{
if(front == MAX-1)
front = 0;
else
front = front+1;
}
}
void display()
{
int front_pos = front,rear_pos = rear;
if(front == -1)
{
printf("Queue is empty\n");
return;
}
printf("Queue elements :");
if( front_pos <= rear_pos )
while(front_pos <= rear_pos)
{
printf("%d ",cqueue_arr[front_pos]);
front_pos++;
}
else
{
while(front_pos <= MAX-1){
printf("%d ",cqueue_arr[front_pos]);
front_pos++;
}
front_pos = 0;
while(front_pos <= rear_pos)
{
printf("%d ",cqueue_arr[front_pos]);
front_pos++;
}
}
printf("\n");
}
int main()
{
int choice,item;
do
{
printf("1.Insert \n");
printf("2.Delete \n");
printf("3.Display \n");
printf("4.Quit \n");
printf("\n");
printf("Enter your choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1 :
printf("Input element for insertion in queue : ");
scanf("%d", &item);
insert(item);
break;
case 2 :
deletion();
break;
case 3:
display();
break;
case 4:
break;
default:
```

```
/tmp/HyUGNJ4yki.o
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 1
Input element for insertion in queue : 4
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input element for insertion in queue : 6
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 3
Queue elements :4 6
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 2
Element deleted from queue is : 4
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 2
Element deleted from queue is : 6
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 2
Queue Underflown
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 4
```