```c
/*
6. Design, Develop and Implement a menu driven Program in C
for the following operations on Circular QUEUE of Characters
a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate Overflow and Underflow situations on Circular QUEUE
d. Display the status of Circular QUEUE
e. Exit
*/

/*
Initially front is initialised to Zero & rear is initialised to -1 indicating queue is empty
We are going to create 3 functions- insert, delete, display.
(front == 0 && rear == MAX-1) means front is pointing at first element and rear is pointing at last element.
(front>0 && rear == front-1)
(front==0)&&(rear==-1)) indicates that queue is empty.
*/
#include <stdio.h>
#include <stdlib.h>

#define MAX 5 // max size of queue is 5, can store 5 elements.
int Q[5];

int front=0;
int rear=-1;

void main()
{
    void insert();  //Insert function
    void delete();  //Delete funcion
    void dispay();  //Display function

    int ch;

    printf("Circlar Queue operations\n");

    printf("\n 1.Insert\n 2.Delete\n 3.Display\n 4.Exit\n");

    while(1)     //The loop keeps on executing recursively based on the choices.
    {
        printf("Enter your choice:\n");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1: insert();
                    break;
            case 2: delete();
                    break;
            case 3: display();
                    break;
            case 4: exit(1);

            default: printf("Invalid option\n");

        }
    }

} //end of main function.


//Defining Insert Function
void insert()
{
```

```
65      int x; //element to be inserted.
66      if((front == 0 && rear == MAX-1)||(front>0 && rear == front-1)) //Checking for
Overflow condition
67
68          printf("Queue overflow\n"); //if both the conditions are true
69
70      else
71          {
72          printf("Enter the element to be inserted\n"); //If Queue is not full, we
insert values.
73          scanf("%d",&x);
74          }
75
76          if(front>0 && rear == MAX-1)
77          {
78              rear=0;
79              Q[rear]=x; //inserting the element at Q[0] position.
80          }
81
82          else
83          {
84              if((front == 0 && rear==-1)||(rear!= front-1))
85              Q[++rear]=x; //increment rear and place the element pointed by rear
86          }
87  } //end of insert function.
88
89
90  //Defining Delete Function
91  void delete()
92  {
93      int a;
94      if((front == 0)&&(rear == -1)) //queue is empty.
95      {
96          printf("Queue underflow\n");
97          exit(1);
98      }
99
100     if(front == rear) //both front and rear pointing to same location.
101     {
102         a=Q[front];
103         rear=-1;
104         front=0;
105     }
106
107     else if(front == MAX-1)
108     {
109         a=Q[front];
110         front=0;
111     }
112
113     else
114         a=Q[front++];
115
116     printf("Deleted element is %d\n",a);
117 }//end of delete function.
118
119 //Defining Display Function
120 void display()
121 {
122     int i,j; //i is used with rear and j is used with front end.
123
124     if(front == 0 && rear == -1) //queue is empty.
125     {
126         printf("Queue doesnt have any element\n");
127         exit(1);
128     }
```

```
129
130     if(front>rear) //Some elements are already inserted in queue
131     {
132         for(i=0;i<=rear;i++)
133             printf("\t%d",Q[i]);
134
135 /*
136 starting from first element (i=0) upto last element inserted which is pointed by
rear (i<=rear),
137 we need to display all the elements.
138 */
139         for(j=front;j<=MAX-1;j++)
140             printf("\t%d",Q[j]);
141 /*
142 starting from element pointed by front(j=front) upto last element(MAX-1),
143 we need to display all the elements.
144 */
145         printf("\n front is at %d position\n", Q[front]);
146         printf("\n Rear is at %d position\n", Q[rear]);
147     }
148
149     else
150     {
151         for(i=front;i<=rear;i++)
152             printf("\t%d",Q[i]);
153
154         printf("\n front is at %d position\n", Q[front]);
155         printf("\n Rear is at %d position\n", Q[rear]);
156     }
157 } //end of display function.
```

OUTPUT:

```
Circlar Queue operations

 1.Insert
 2.Delete
 3.Display
 4.Exit
Enter your choice:
2
Queue underflow


 1.Insert
 2.Delete
 3.Display
 4.Exit
Enter your choice:
3
Queue doesnt have any element.


  1.Insert
 2.Delete
 3.Display
 4.Exit
Enter your choice:
1
Enter the element to be inserted
10
Enter your choice:
1
Enter the element to be inserted
20
Enter your choice:
1
Enter the element to be inserted
30
Enter your choice:
1
Enter the element to be inserted
40
Enter your choice:
1
Enter the element to be inserted
50
Enter your choice:
1
Queue overflow
```

```
Enter your choice:
3
        10      20      30      40      50
 front is at 10 position

 Rear is at 50 position


Enter your choice:
2
Deleted element is 10
Enter your choice:
2
Deleted element is 20
Enter your choice:
2
Deleted element is 30
Enter your choice:
2
Deleted element is 40
Enter your choice:
3
        50
 front is at 50 position

 Rear is at 50 position
Enter your choice:
1
Enter the element to be inserted
60
Enter your choice:
1
Enter the element to be inserted
70
Enter your choice:
1
Enter the element to be inserted
80
Enter your choice:
3
        60      70      80      50
 front is at 50 position

 Rear is at 80 position


Enter your choice:
2
Deleted element is 50
Enter your choice:
```

```
2
Deleted element is 60
Enter your choice:
2
Deleted element is 70
Enter your choice:
2
Deleted element is 80
Enter your choice:
2
Queue underflow
```