Circular Queue implementation:-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#define max 30
 typedef struct
  {
	int s[max];
	int rear, front;
  }queue;


 int isFull(queue *q)
 {
	if(q->front==(q->rear+1)%max)
	{
		return 1;
	}
	else
	{
		return 0;
	}
 }

  int isEmpty(queue *q)
 {
	if(q->rear==-1)
	{
		return 1;
	}
	else
	{
		return 0;
	}
 }

 void enqueue(queue *q,int a)
 {
	if(isFull(q))
	{
		printf("****************************\n");
		printf("Queue is full\n");
		printf("****************************\n");
	}
	else
	{
		if(q->front==-1)
		{
			q->front=0;
		}
```

```
        q->rear=(q->rear+1)%max;
        q->s[q->rear]=a;
        printf("Number is successfully queued\n");
        printf("****************************\n");
    }
}

void dequeue(queue *q)
{
    int s1;
    if(isEmpty(q))
    {
        printf("****************************\n");
        printf("Queue is Empty\n");
        printf("****************************\n");
        return;
    }
    else
    {
        s1=q->s[q->front];
        //printf("\n%d %d\n\n",q->front,q->rear);
            if(q->front==q->rear)
        {
            q->front=q->rear=-1;
        }
        q->front=(q->front+1)%max;
        //printf("\n%d %d\n\n",q->front,q->rear);
        printf("%d",s1);
        return;
    }
}

void display(queue *q)
{
    int i;
    //printf("%d",);
    printf("\n");
    printf("****************************\n");
    printf("Numbers in queue\n");
    i=q->front;
    while(1)
    {
        printf("%d  %d\n",q->s[i],i);
        if(i==q->rear)
        {
            break;
        }
        i=(i+1)%max;

    }
    printf("\n****************************\n");
}
```

```
int main(void)
{
	queue q;
	int a,n;

	q.front=q.rear=-1;

	while(1)
	{
		printf("\nCircular queue program");
		printf("\n1 for adding\n2 for removing\n");
		printf("3 for display\n");
		printf("4 for clearscreen\n5 for exit program\n\n");
		scanf("%d",&a);
		switch(a)
		{
			case 1:{
				printf("Enter the number to be queued");
				scanf("%d",&n);
				enqueue(&q, n);
				break;
				}
			case 2:{dequeue(&q);break;}
			case 3:{display(&q);break;}
			case 4:{clrscr();break;}
			case 5:{printf("Thank You");exit(0);break;}
		}
	}

}
```

Output:-

```
Circular queue program
1 for adding
2 for removing
3 for display
4 for clearscreen
5 for exit program

1
Enter the number to be queued:
55
***************************
Number is successfully queued
***************************

Circular queue program
1 for adding
2 for removing
3 for display
4 for clearscreen
5 for exit program

1
Enter the number to be queued:
56
***************************
Number is successfully queued
***************************

Circular queue program
1 for adding
2 for removing
3 for display
4 for clearscreen
5 for exit program

1
Enter the number to be queued:
57
***************************
Number is successfully queued
***************************

Circular queue program
1 for adding
2 for removing
3 for display
4 for clearscreen
5 for exit program

1
Enter the number to be queued:
```

```
58
***************************
Number is successfully queued
***************************

Circular queue program
1 for adding
2 for removing
3 for display
4 for clearscreen
5 for exit program

3

***************************
Numbers in queue
55  0
56  1
57  2
58  3

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

Circular queue program
1 for adding
2 for removing
3 for display
4 for clearscreen
5 for exit program

2
***************************
55 is successfully dequeued
***************************

Circular queue program
1 for adding
2 for removing
3 for display
4 for clearscreen
5 for exit program

5
Thank You
```