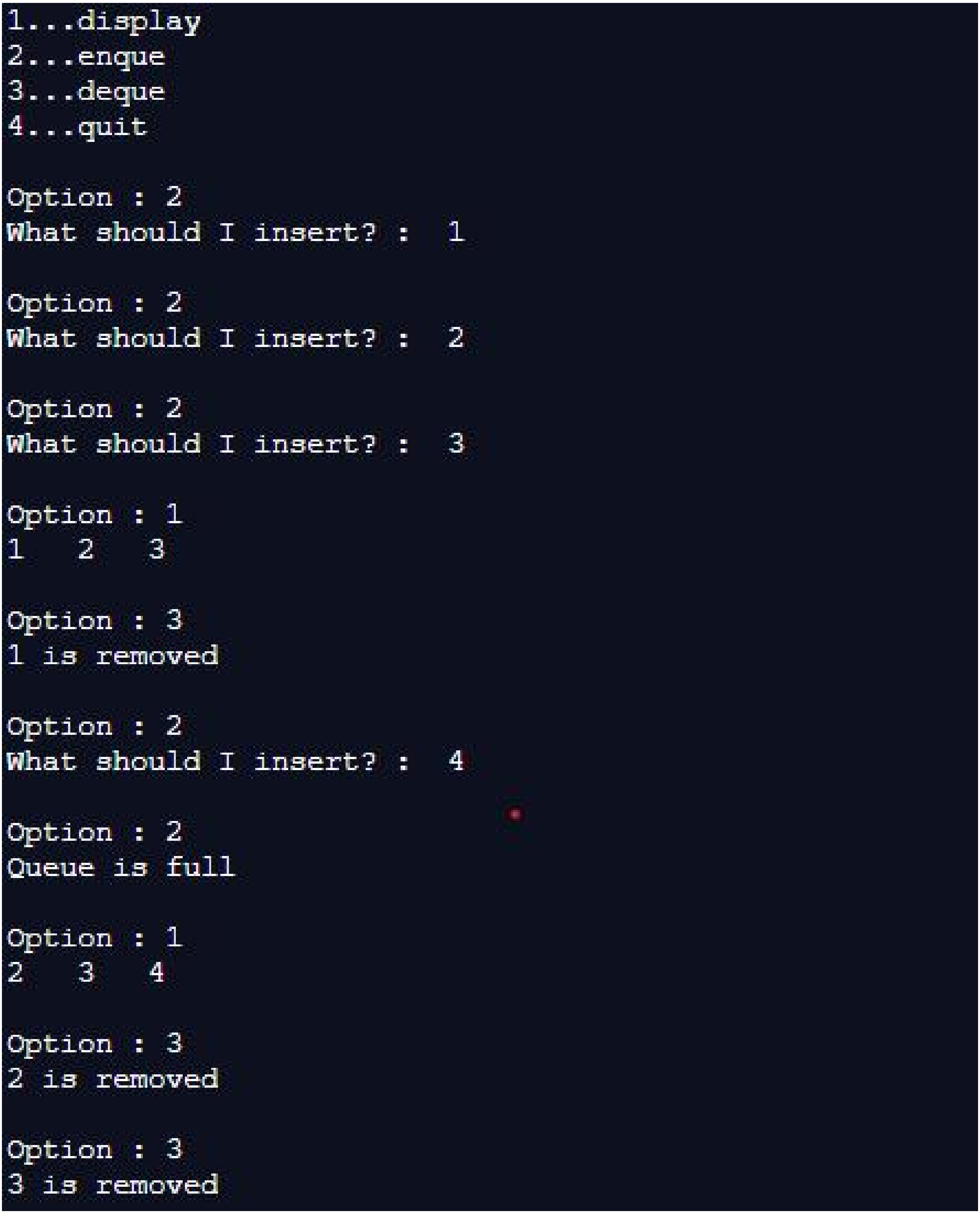
```
#include <stdio.h>

#define n 3
int que[n] ,front=-1 ,rear=-1 ;


void enqueue(int que[] ,int* front ,int* rear ){
        if( (*rear+1)%n == *front)
        {
                printf("Queue is full\n");
                return;
        }
        if( *front ==-1 )
        {
                *front=(*front+1)%n;
        }
        int item;
        printf("What should I insert? :  ");
    scanf("%d",&item);

        *rear=(*rear+1)%n;
        que[*rear]=item;

        //printf("**%d,%d**",*front,*rear);
}

void dequeue(int que[] ,int* front ,int* rear ){
        int item;
        if( *front==-1 && *rear==-1 ){
            printf("Empty Queue\n");
        }
        else{
        item=que[*front];
                if( *front==*rear ){
                        *rear=-1;
                        *front=-1;
                }
                else{
                //que[*front]='\0';
                        *front=(*front+1)%n;
                }
        printf("%d is removed\n",item);
        }
        //printf("**%d,%d**",*front,*rear);
}

void display(int que[] ,int front ,int rear){
        if( front==-1 && rear==-1 ){
                printf("Queue is empty\n");
        }
        else
        {
            int i=front;
            if((i+1)%n==(rear+n-1)%n)
            {
                while((i+1)%n!=rear)
                {
                                printf("%d\t",que[i]);
                                if(i==(front+n-1)%n)
                                    break;
                                i=(i+1)%n;
                        }
            }
                while((i+1)%n!=(rear+n-1)%n)
                {
                        printf("%d\t",que[i]);
                        if(i==(front+n-1)%n)
                            break;
                        i=(i+1)%n;
                }
                printf("\n");
        }
        //printf("**%d,%d**",front,rear);
}

int main(){
        //printf("Enter the total size of the Queue : ");
    //int n=3;
    //scanf("%d", & n);
    int choice;
    int item=0;
    printf("1...display\n");
    printf("2...enque\n");
    printf("3...deque\n");
    printf("4...quit\n");

    int quit=1;
    while(quit!=0){
        printf("\nOption : ");
        scanf("%d",&choice);
        switch(choice){
                case 1: display(que,front,rear);
                        break;
                case 2: enqueue(que,&front,&rear);
                        break;
                case 3: dequeue(que,&front,&rear);
                        break;
                case 4: quit=0;
                    printf("*****Program aborted*****");
        }
    }
    return 0;
}
```



```
1...display
2...enque
3...deque
4...quit

Option : 2
What should I insert? :  1

Option : 2
What should I insert? :  2

Option : 2
What should I insert? :  3

Option : 1
1   2   3

Option : 3
1 is removed

Option : 2
What should I insert? :  4

Option : 2
Queue is full

Option : 1
2   3   4

Option : 3
2 is removed

Option : 3
3 is removed
```

8= Circular Queue

AIM

To impliment Circular Queue Data Structure

ALGORITHM

ENQUEUE (ITEM)

```
1.  START
2.  If (FRONT = = ((REAR+1) % size) then
        1. Print "CQ Full"
3   Else
        1. If (FRONT = = REAR = = -1) then
               1. FRONT = 0
               2. REAR = 0
               3. CQ[REAR] = ITEM
        2. Else
               1. REAR = ((REAR+1) % size)
               2. CQ[REAR] = ITEM
        3. End If
4. End If
5. STOP
```

DEQUEUE ( )

```
1. START
2. If (FRONT == REAR == -1)
       1. print "CQ Empty"
3. Else
       1. ITEM = CQ [FRONT]
       2. If (FRONT == REAR)
              1. FRONT = -1
              2. REAR = -1
       3. Else
              1. FRONT = (FRONT + 1) % size
       4 End If.
4. End If
5. STOP.
```