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

struct queue
{
    int size;
    int f;
    int r;
    int* arr;
};


int isEmpty(struct queue *q){
    if(q->r==q->f){
        return 1;
    }
    return 0;
}

int isFull(struct queue *q){
    if(q->r==q->size-1){
        return 1;
    }
    return 0;
}

void enqueue(struct queue *q, int val){
    if(isFull(q)){
        printf("This Queue is full\n");
    }
    else{
        q->r++;
        q->arr[q->r] = val;
        printf("Enqued element: %d\n", val);
    }
}

int dequeue(struct queue *q){
    int a = -1;
```

```
40:     if(isEmpty(q)){
41:         printf("This Queue is empty\n");
42:     }
43:     else{
44:         q->f++;
45:         a = q->arr[q->f];
46:     }
47:     return a;
48: }
49:
50: int main(){
51:     struct queue q;
52:     q.size = 4;
53:     q.f = q.r = 0;
54:     q.arr = (int*) malloc(q.size*sizeof(int));
55:
56:     // Enqueue few elements
57:     enqueue(&q, 12);
58:     enqueue(&q, 15);
59:     enqueue(&q, 1);
60:     printf("Dequeuing element %d\n", dequeue(&q));
61:     printf("Dequeuing element %d\n", dequeue(&q));
62:     printf("Dequeuing element %d\n", dequeue(&q));
63:     enqueue(&q, 45);
64:     enqueue(&q, 45);
65:     enqueue(&q, 45);
66:
67:     if(isEmpty(&q)){
68:         printf("Queue is empty\n");
69:     }
70:     if(isFull(&q)){
71:         printf("Queue is full\n");
72:     }
73:
74:     return 0;
75: }
76:
```