Q.Queue using array.

```
#include <stdio.h>
#define MAX_SIZE 100
struct Queue {
    int items[MAX_SIZE];
    int front;
    int rear;
};
void enqueue(struct Queue *q, int value) {
    if (q->rear == MAX_SIZE - 1) {
        printf("Queue is full");
    } else {
        if (q->front == -1) {
            q->front = 0;
        }
        q->rear++;
        q->items[q->rear] = value;
    }
}
int dequeue(struct Queue *q) {
    int item;
    if (q->front == -1) {
        printf("Queue is empty");
        return -1;
    } else {
        item = q->items[q->front];
        q->front++;
        if (q->front > q->rear) {
            q->front = q->rear = -1;
        }
        return item;
    }
}
int main() {
    struct Queue q;
    q.front = -1;
    q.rear = -1;
    enqueue(&q, 10);
    enqueue(&q, 20);
    enqueue(&q, 30);
    printf("%d dequeued from the queue\n", dequeue(&q));
    printf("%d dequeued from the queue\n", dequeue(&q));
    return 0;
}
```

Output:

10 dequeued from the queue
20 dequeued from the queue

Q.Queue using Linear System.

```
#include <stdio.h>
#define MAX_SIZE 100
int queue[MAX_SIZE];
int front = -1, rear = -1;
void enqueue(int value) {
    if (rear == MAX_SIZE - 1) {
        printf("Queue is full.\n");
    } else {
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
    }
}
int dequeue() {
    int value;
    if (front == -1) {
        printf("Queue is empty.\n");
        return -1;
    } else {
        value = queue[front];
        front++;
        if (front > rear) {
            front = rear = -1;
        }
        return value;
    }
}
int main() {
    enqueue(10);
    enqueue(20);
    enqueue(30);
    printf("Dequeued: %d\n", dequeue());
    printf("Dequeued: %d\n", dequeue());
```

```
    printf("Dequeued: %d\n", dequeue());
    printf("Dequeued: %d\n", dequeue());
    return 0;
}
```

Output:

```
Dequeued: 10
Dequeued: 20
Dequeued: 30
Queue is empty.
Dequeued: -1
```

Q.Queue using Array.

```
#include <stdio.h>
#define MAX_SIZE 100
struct Queue {
    int items[MAX_SIZE];
    int front;
    int rear;
};
void enqueue(struct Queue *q, int value) {
    if (q->rear == MAX_SIZE - 1) {
        printf("Queue is full");
    } else {
        if (q->front == -1) {
            q->front = 0;
        }
        q->rear++;
        q->items[q->rear] = value;
    }
}
int dequeue(struct Queue *q) {
    int item;
    if (q->front == -1) {
        printf("Queue is empty");
        return -1;
    } else {
        item = q->items[q->front];
        q->front++;
        if (q->front > q->rear) {
            q->front = q->rear = -1;
```

```
        }
        return item;
    }
}
int main() {
    struct Queue q;
    q.front = -1;
    q.rear = -1;
    enqueue(&q, 10);
    enqueue(&q, 20);
    enqueue(&q, 30);
    printf("%d dequeued from the queue\n", dequeue(&q));
    printf("%d dequeued from the queue\n", dequeue(&q));
    return 0;
}
```

Output:

```
10 dequeued from the queue
20 dequeued from the queue
```