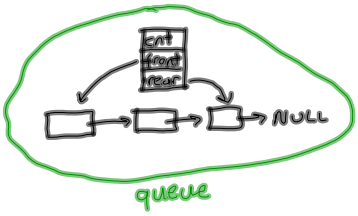
Queues

The basic operations on a queue (FIFO list) are enqueue which inserts an element at the end of the queue (called rear), and dequeue which deletes (and returns) the element at the start of the queue (known as front).

Implementations of Queue data structure

1) Linked List Implementation




```
#define QUEUE_SIZE 10
typedef struct {
    int cnt;
    struct node *front;
    struct node *rear;
} queue;
```

Queue Operations

a) Initialization

```
void initialize (queue *q){
    q->cnt = 0;
    q->front = q->rear = NULL;
}
```

b)

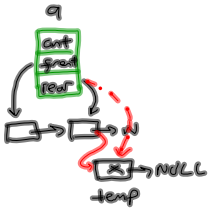
```
int isfull ( queue *q){
    if ( q->cnt == QUEUE_SIZE)
        return 1;
    else
        return 0;
}
```

c)

```
int isempty ( queue *q){
    if ( q->cnt == 0)
        return 1;
    else
        return 0;
}
```

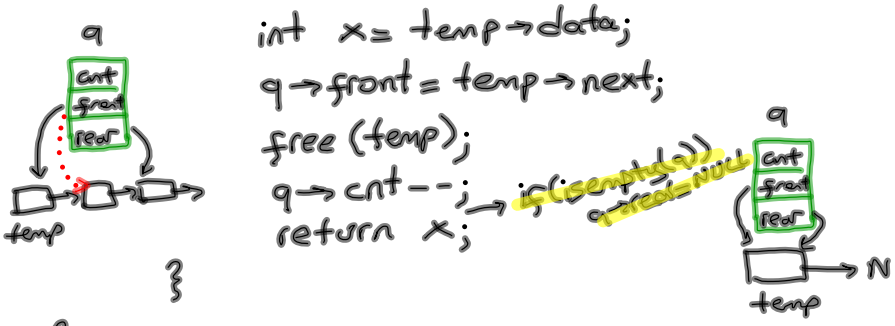
d) Enqueue (add an element at the end)

```
void enqueue ( queue *q, int x){
    if ( isfull(q))
        printf ("Queue is full");
    else{
        struct node *temp= ... malloc...
        temp->data= x;
        temp->next= NULL;
        if ( isempty(q))
            q->front= q->rear=temp;
        else {
            q->rear->next=temp;
            q->rear= temp
        }
        q->cnt++;
    }
}
```

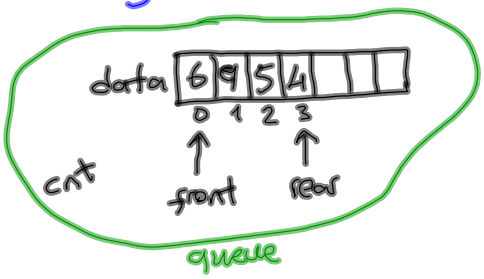


e) Dequeue (deletes and returns the first element)

```
int dequeue ( queue *q){
    if ( isempty(q)){
        printf ("Queue is empty");
        return -100;
    }else{
        struct node *temp= q->front;
        int x= temp->data;
        q->front = temp->next;
        free (temp);
        q-> cnt--;
        return x;
    }
}
```



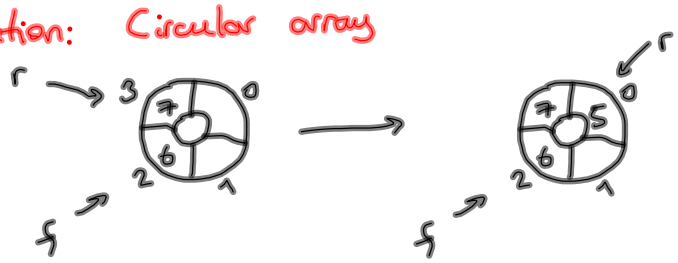
2. Array Implementation of queues




```
typedef struct{
    int front, rear;
    int cnt;
    int data[QUEUE_SIZE];
}queue;
```

Problem: After several enqueue and dequeu ops.

Solution: Circular array

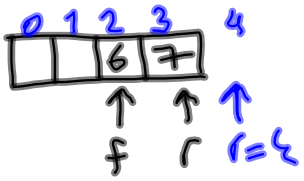


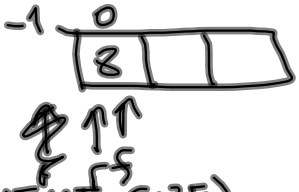
a) Initialization

```
void initialize ( queue *q){
    q-> cnt= 0;
    q-> front= 0;
    q-> rear= -1;
}
```

b) Enqueue

```
void enqueue ( queue *q, int x){
    if ( isfull(q))
        printf ("Queue is full");
    else{
        q->rear++;
        q->cnt++;
        if ( q->rear== QUEUE_SIZE)
            q->rear==0
        q->data[q->rear] = x;
    }
}
```





c) Dequeue

```
int dequeue ( queue *q){
    if ( isempty(q)){
        printf ("Queue is empty");
        return -100;
    }else{
        int x=q->data[q->front];
        q->front++;
        q->cnt--;
        if( q->front== QUEUE_SIZE)
            q->front= 0;
        return x;
}
```