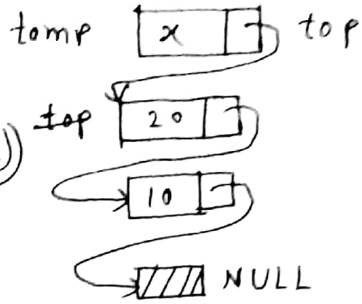
Linked Implementation of stack:

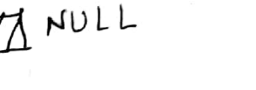


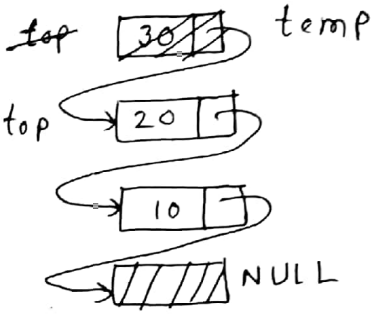
push(stack ⇐ x)

```
temp=(node*)malloc(sizeof(node));
temp->data = x;
temp->next = top;
top = temp;
```

pop(stack ⇒ x)

```
if (top == NULL)
    printf("stack empty");
else {
    temp= top;
    top = top->next;
    free(temp);
}
```





Queue

A Queue is a FIFO (First In First out) data structure

Queue has two operations

- Insert (To insert something into Queue) → rear
- Delete (To remove " from Queue) → Front

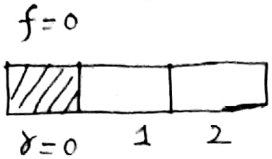
Array implementation of Queue

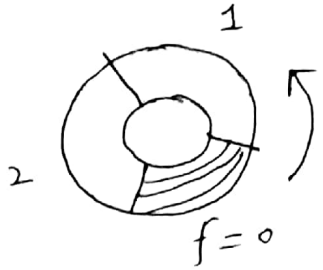
(2) Show Queue status for each of the following Queue operations Assuming Queue size, m = 2 (Array)

Insert(10), Insert(20), Delete(), Insert(30), Insert(40), Delete(), Delete(), Delete()

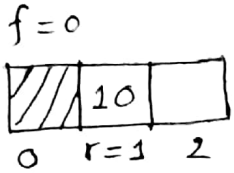
Ans:

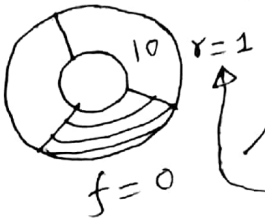
Initial:





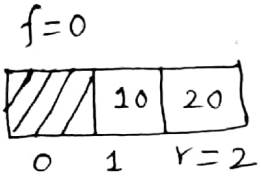
Insert(10):

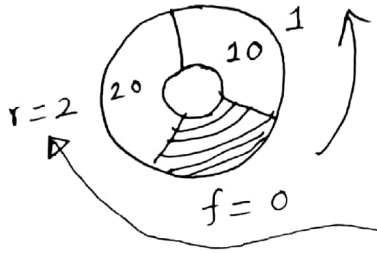




$s = (r+1) \bmod (m+1)$
$= (0+1) \bmod (2+1)$
$= 1 \bmod 3 = 1$

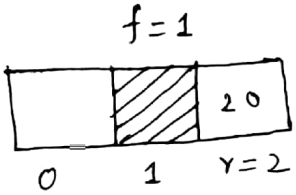
Insert(20):

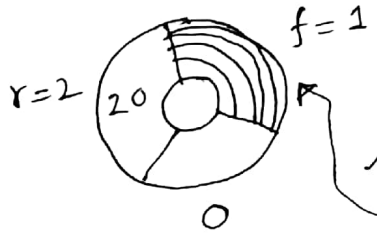




$s = (r+1) \bmod (m+1)$
$= (1+1) \bmod (2+1)$
$= 2 \bmod 3 = 2$

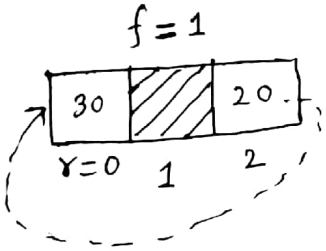
Delete():

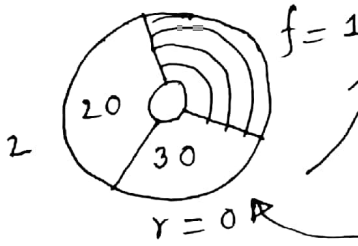




$f = (f+1) \bmod (m+1)$
$= (0+1) \bmod (2+1)$
$= 1 \bmod 3 = 1$

Insert(30):





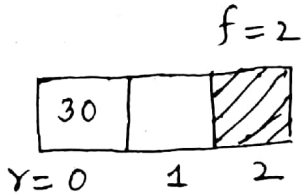
$s = (r+1) \bmod (m+1)$
$= (2+1) \bmod (2+1)$
$= 3 \bmod 3 = 0$

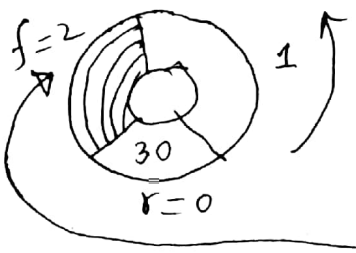
Insert(40): "Queue full"

"Queue full"

$s = (r+1) \bmod (m+1)$
$= (0+1) \bmod (2+1)$
$= 1 \bmod 3 = 1$
$= f$

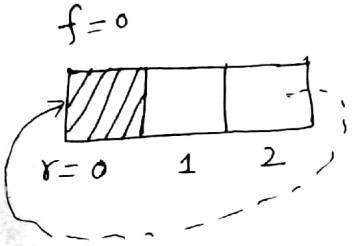
Delete():

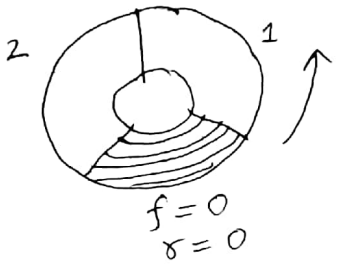




$f = (f+1) \bmod (m+1)$
$= (1+1) \bmod (2+1)$
$= 2$

Delete():





Delete(): "Queue empty"

"Queue empty"

Logic

Insert(Queue ⟸ x)

```
s = (r+1) mod (m+1)
if (s==f)
    printf("Queue Full");
else{
    Queue[s] = x;
    r=s;
}
```

Delete(Queue ⟹ x)

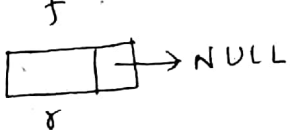
```
if (f==r)
    printf("Queue Empty");
else {
    f = (f+1) mod (m+1);
    Queue[f] = NULL;
}
```

Linked Implementation of Queue

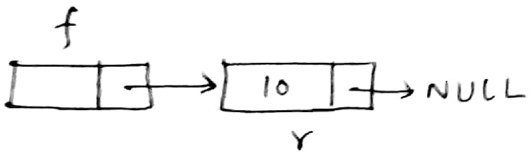
show Queue status for each of the following Queue operations
Insert(10), Insert(20), Delete(), Insert(30), Insert(40),
Delete(), Delete(), Delete()

Ans:

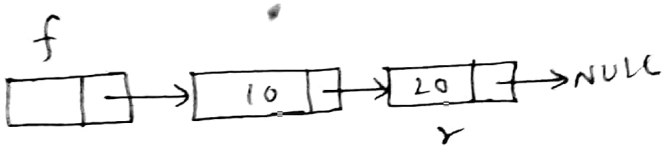
Initial:



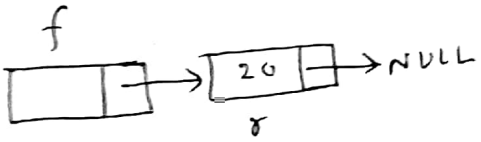
Insert (10):



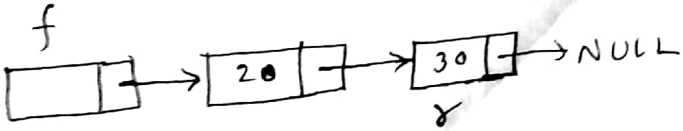
Insert (20):



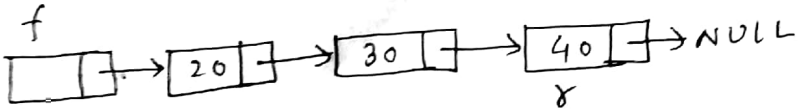
Delete():



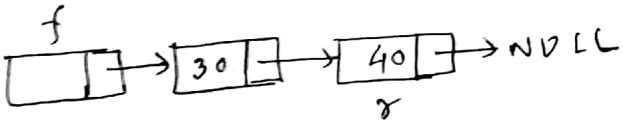
Insert (30):



Insert (40):

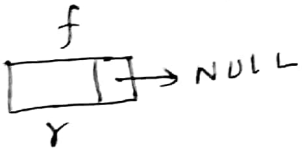


Delete ():



Delete():

Delete():



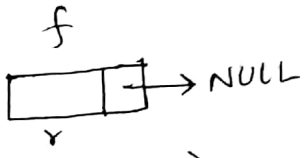
Delete(): "Queue Empty"

Logic

Insert

```
temp=(node*) malloc (sizeof(node));
temp->data = x;
r->next = temp;
temp->next = NULL;
r= temp;
```

Delete()




```
if (f==r)
  printf("Queue Empty");
else {
temp= f->next;
f->next = temp->next;
free(temp);
if (f->next==NULL)
      r=f;
}
```