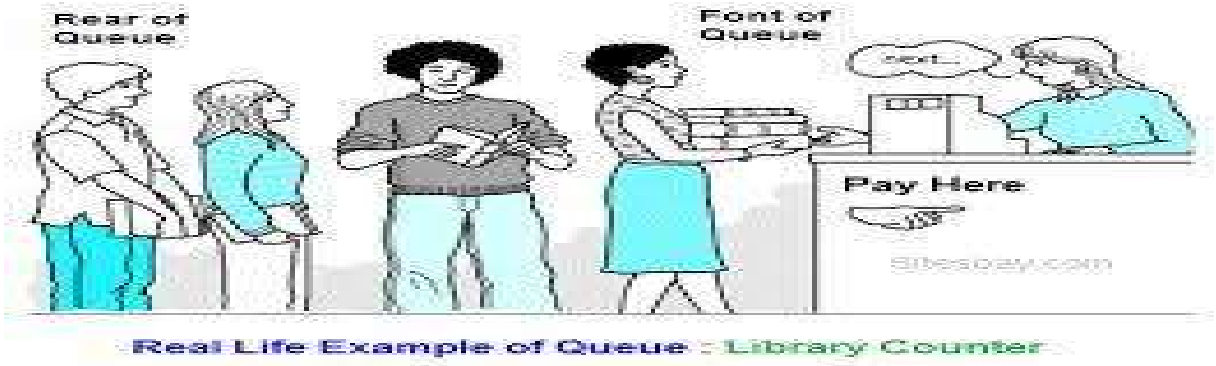
QUEUE

A queue is linear data structure and collection of elements. A queue is another special kind of list, where items are inserted at one end called the rear and deleted at the other end called the front. The principle of queue is a “FIFO” or “First-in-first-out”.

Queue is an abstract data structure. A queue is a useful data structure in programming. It is similar to the ticket queue outside a cinema hall, where the first person entering the queue is the first person who gets the ticket.

A real-world example of queue can be a single-lane one-way road, where the vehicle enters first, exits first.

More real-world examples can be seen as queues at the ticket windows and bus-stops and our college library.



Basic Operations in Queue

Queue operations also include initialization of a queue, usage and permanently deleting the data from the memory.

The most fundamental operations in the queue ADT include: enqueue(), dequeue(), peek(), isFull(), isEmpty(). These are all built-in operations to carry out data manipulation and to check the status of the queue.

Queue uses two pointers – front and rear. The front pointer accesses the data from the front end (helping in enqueueing) while the rear pointer accesses data from the rear end (helping in dequeuing).

1) Algorithm to insert an elememt a static queue:

```
Step – 1 : [Check overflow condition]
                    if rear <size-1 then

step – 2 : [Increment rear pointer]
                    rear = rear+1

step – 3 : [Insert an element]
                    q[rear] = value

step – 4 : [set the front pointer]
                    if front = -1 then
                          front = 1
step-5   : Else
                    output "Queue overflow"

step – 5 : exit
```

2) Algorithm to Delete an element in a static queue:

```
Step – 1 : [check Underflow condition]
                    if front = -1 then
                          output "Queue Underflow"
                          exit
step – 2 : [Remove an element]
                    value = q[front]
step – 3 : [check for empty queue]
                    if front = rear then
                          front = -1
                          rear = -1
                    else
                          front = front+1

step – 4 : return (value)
```

Program for static queue

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int front = -1;
int rear = -1;
int q[SIZE];

void insert();
void del();
void display();

int main() {
    int choice;

    do {
        printf("\n\t Menu\n");
        printf("1.Insert\n");
        printf("2. Delete\n");
        printf("3.Display\n");
        printf("4. Exit\n");

        printf("Enter Your Choice: ");
```

```
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            insert();
            display();
            break;
        case 2:
            del();
            display();
            break;
        case 3:
            display();
            break;
        case 4:
            printf("End of Program\n");
            exit(0);
        default:
            printf("Invalid choice!\n");
    }
} while (choice != 4);

    return 0;
```

```
}

void insert() {
    int no;
    printf("Enter Number: ");
    scanf("%d", &no);

    if (rear == SIZE - 1) {
        printf("Queue overflow\n");
        return;
    }
    if (front == -1)
        front = 0;
    q[++rear] = no;
}

void del() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow\n");
        return;
    } else {
        printf("Element deleted: --> %d\n", q[front]);
        if (front == rear) {
```

```
            front = -1;
            rear = -1;
        } else {
            front++;
        }
    }
}

void display() {
    int i;
    if (front == -1) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements: ");
    for (i = front; i <= rear; i++)
        printf("%d ", q[i]);
    printf("\n");
}
```

Output :-

```
        Menu
1.Insert
2. Delete
3.Display
4. Exit
Enter Your Choice: 1
Enter Number: 56
Queue elements: 56

        Menu
1.Insert
2. Delete
3.Display
4. Exit
Enter Your Choice: 1
Enter Number: 98
Queue elements: 56 98

        Menu
1.Insert
2. Delete
3.Display
4. Exit
Enter Your Choice: 2
Element deleted: --> 56
Queue elements: 98

        Menu
1.Insert
2. Delete
3.Display
4. Exit
Enter Your Choice: 3
Queue elements: 98

        Menu
1.Insert
2. Delete
3.Display
4. Exit
Enter Your Choice: 4
End of Program


...Program finished with exit code 0
Press ENTER to exit console.
```

Circular Queue:

- The problem with simple queue is that once we insert the elements and when we are removing elements using front pointer these elements becomes blank. As the elements become blank we cannot insert any elements in that blank space.
- So to overcome this problem there is technique available known as circular queue. In this technique when rear reaches the queue's size the first element will become the queue's new rear

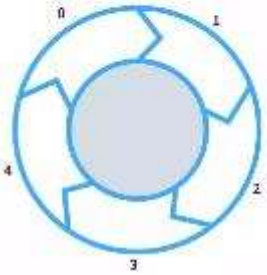
- **NOTE:** When the queue is full, the first element of queue becomes the rear if and only if front has moved forward otherwise it will be again in a "queue full(overflow)" state.

EXAMPLE:

Size of Circular Queur Q is 5

Initially, Q is empty and the queur pointers
F (frant) and R (rear) are initialized to -1

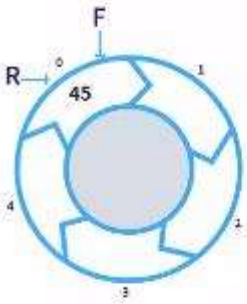
F = R = -1



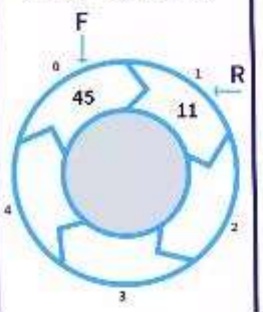
Tasks to perform :-

(i) Insert 45, 11, 23, 81, 57.
(ii) Delete 45, 11, 23, 81.
(iii) Insert 29.
(iv) Delete 57, 29.
(v) Insert 17, 19.

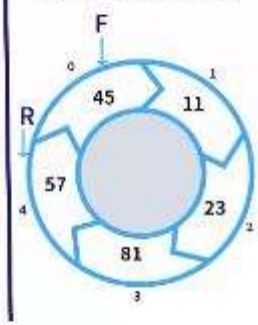
(i) Insert 45:
F=R= 0 Q[R]= 45



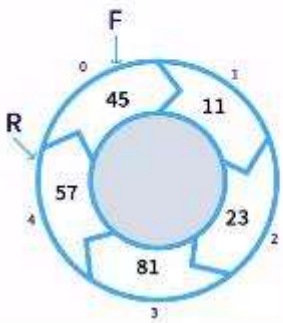
Insert 11:
R=R+1 Q[R+1]= 11



Insert 23,81,57:
R=R+1 Q[R+1]= Val

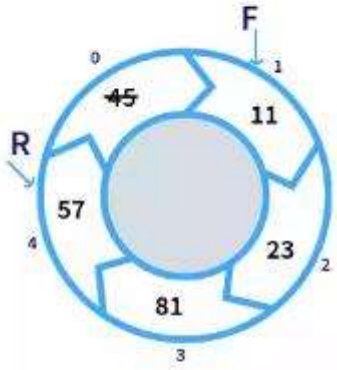


(ii) Insurt 32:
(R+1)%n==F
Overflow condition is
reached Cannot insert 32



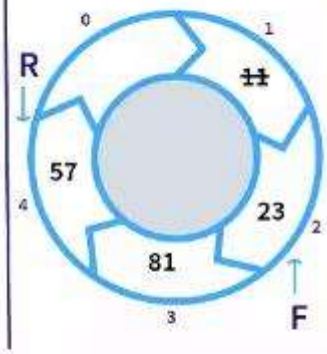
(iii) Delete 45:

F=(f+1) % n



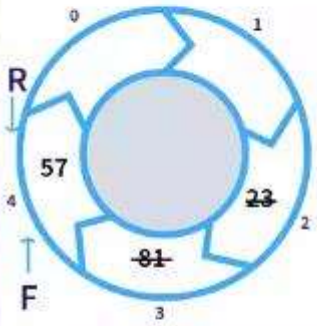
Delete 11:

F=(F+1) % n



Delete 23,81:

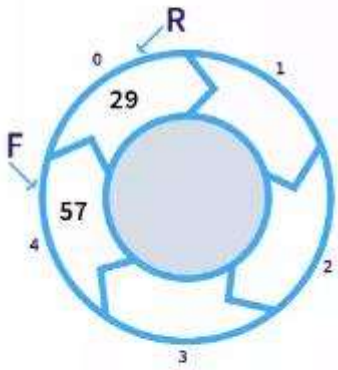
F=(F+1) % n



(iv) Insert 29:

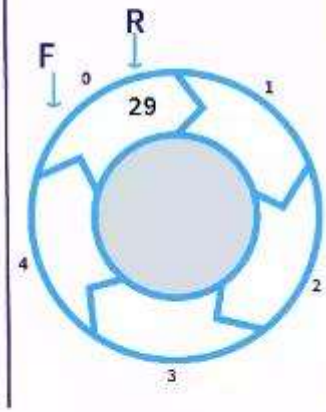
Q [(R+1)%n] = 29

R=(R+1) % n



(v) Delete 57:

F=(F+1) % n

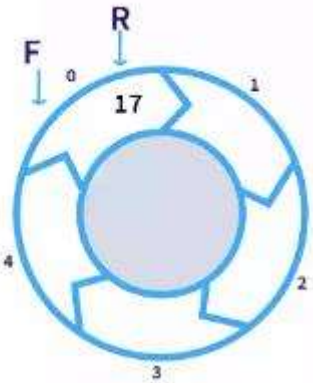


Delete 29:

Queue becomes empty after deleting 29. Any further deletion will lead to underflow condition. The (*) you see below it means that you have to put the image in this screenshot marked with (*) and F=R=-1

(vi) Enqueue 57:

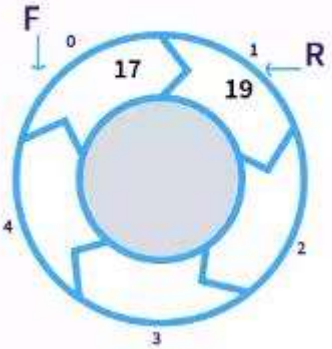
F= R = 0



Enqueue 17:

After the last Dequeue, the circular queue becomes empty and insertion starts from 0th position again.

Enqueue 19:



Circular queue

Algorithm : Insertion in a queue

```
1 : [Check for overflow]
        if front=0 and rear=size-1
            Output : " Queue is Overflow"
            return

Step-2:  [check for overflow when rear is less than front]
      else if rear=front-1
                  Output : " Queue is Overflow"
            return

Step-3 : [insert element in queue]
            else if front<0
                  front=0;
                  rear=0;
                 q[rear]=no;

 Step-4 : [Check if rear at the end of  the queue)
           else if rear=size-1
            rear=0;
            q[rear]=no;
 Step-5  : [Insert element]
            else
                   rear=rear+1;
                  if  rear=front
                        Output "Queue is overflow "
                  else
                        q[rear]=no;
 Step-6 :  return
```

Algorithm : Deletion in a queue

```
 Step -1 : [Check for underflow]
            if front<0
                   Output"Underflow"
                  return

 Step-2 : [Remove the element]
            no=q[front];
            q[front]=NULL;
            Output "Element deleted : ",no

Step-3 : [Check whether the queue is empty or not]
            if front=rear
                  front=-1;
                  rear=-1;

Step-4 : [Check for the front pointer position
             else if front = size-1
                    front=0;
            else
                     front=front+1;
```

Step- 5 : Return

Program : //Circular Queue

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
#define size 6

int rear, front;
int no;
int q[size];
/* Function to create queue */
void Insert_q() {
   printf("\n Input the Element : ");
   scanf("%d", &no);
   if ((front == 0 && rear == size - 1) || (rear == front - 1)) {
      printf("\n Queue is Overflow");
      return;
   } else if (front < 0) {
      front = 0;
      rear = 0;
      q[rear] = no;
   } else if (rear == size - 1) {
      rear = 0;
      q[rear] = no;
```

```
    } else {
        rear++;
        if (rear == front) {
            printf("\n Queue is overflow ");
            return;
        } else {
            q[rear] = no;
        }
    }
}
```

/* Function to perform delete operation */

```
void Delete_q() {
    if (front < 0) {
        printf("\n Queue is Underflow");
        return;
    }
    no = q[front];
    q[front] = -1; // Mark the element as deleted
    printf("\n Element deleted : %d", no);
    if (front == rear) {
        front = -1;
        rear = -1;
    } else if (front == size - 1) {
        front = 0;
```

```
    } else {
        front++;
    }
}
/* Output function */
void Display_q() {
    int i;
    if (front < 0) {
        printf("\n Queue is underflow");
        return;
    }
    if (rear >= front) {
        for (i = front; i <= rear; i++) {
            printf("\n q[%d] = %d", i, q[i]);
        }
    } else {
        for (i = front; i < size; i++) {
            printf("\n i-2 q[%d] = %d", i, q[i]);
        }
        for (i = 0; i <= rear; i++) {
            printf("\n i-3 q[%d] = %d", i, q[i]);
        }
    }
}
```

```
/* Function main */
int main() {
   int choice;
   do {
      printf("\n\t Menu");
      printf("\n 1. Insert");
      printf("\n 2. Delete");
      printf("\n 3. Display ");
      printf("\n 4. Exit");
      printf("\n Enter Your Choice : ");
      scanf("%d", &choice);
      switch (choice) {
         case 1:
            Insert_q();
            Display_q();
            break;
         case 2:
            Delete_q();
            Display_q();
            break;
         case 3:
            Display_q();
            break;
         case 4:
```

```
            printf("End of Program");
            break;
        default:
            printf("Invalid choice");
 }
   } while (choice != 4);
return 0;
}
```

Output:-

```
                                                    Menu
 1. Insert
 2. Delete
 3. Display
 4. Exit
 Enter Your Choice : 2

 Element deleted : 0
 Queue is underflow                                 Menu
 1. Insert
 2. Delete
 3. Display
 4. Exit
 Enter Your Choice : 2
```

```
Queue is Underflow

Queue is underflow                    Menu

1. Insert

2. Delete

3. Display

4. Exit

Enter Your Choice :4

...Program finished with exit code 0

Press ENTER to exit console.
```