CIRCULAR QUEUE

In circular queue also operations such as insert and delete are performed based on **FIFO** (first in first out) principle and the last position is connected back to the first position to make a circle. It is also called **'Ring Buffer'**.

Note : 1) In a normal Queue (also known as linear queue), we can insert elements until queue becomes full. But once queue becomes full, we cannot insert the next element even if there is a space in front of queue.

2) But in Circular queue, if any one location / position is empty, then we can insert an element into the queue.

ie., It is possible to insert new elements, if and only if those locations (starting locations) are empty.

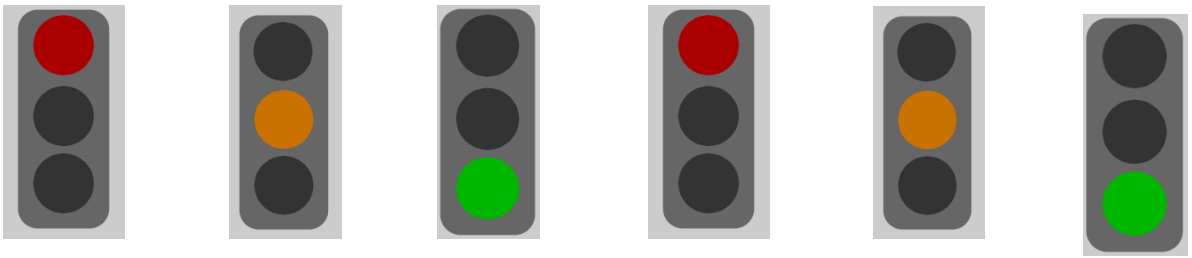
Basic Operations on Circular Queue:

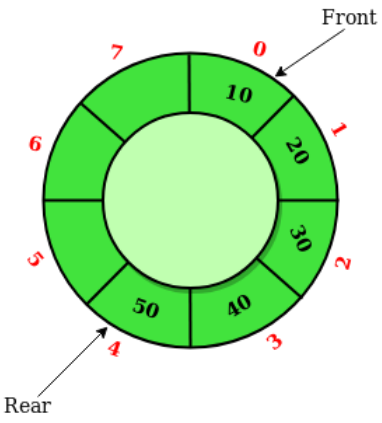
- Enqueue: Add an element to the end of the queue
- Dequeue: Remove an element from the front of the queue
- IsEmpty: Check if the queue is empty
- IsFull: Check if the queue is full
- Peek: Get the value / element of the front of the queue without removing it

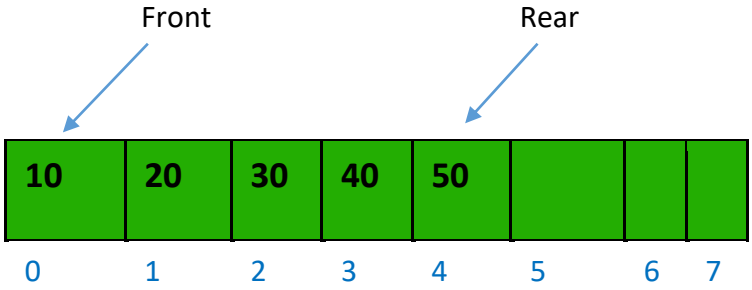
Circular Queue Implementation:

Circular Queue can be implemented in two ways

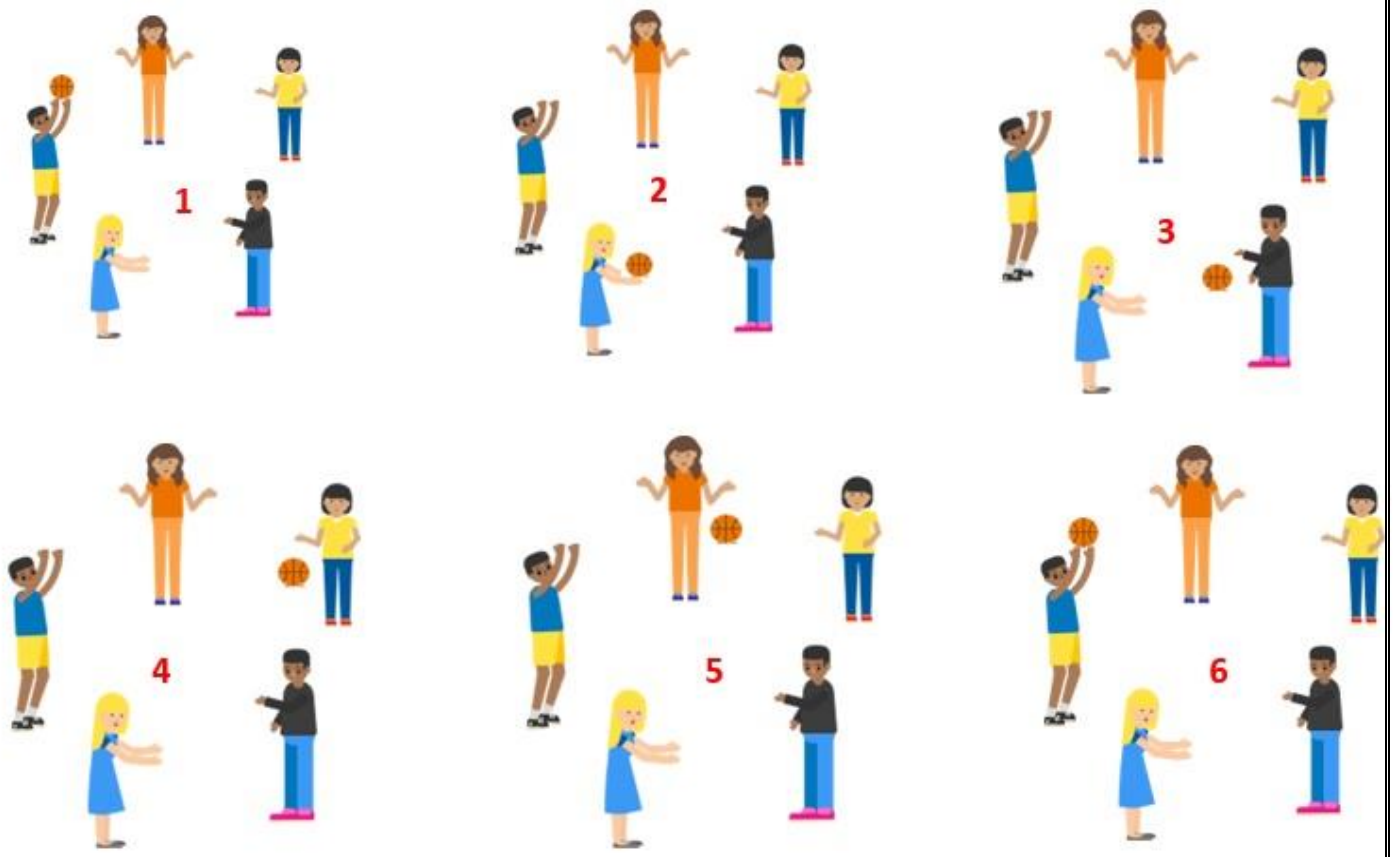
1) Array method 2) Linked list method

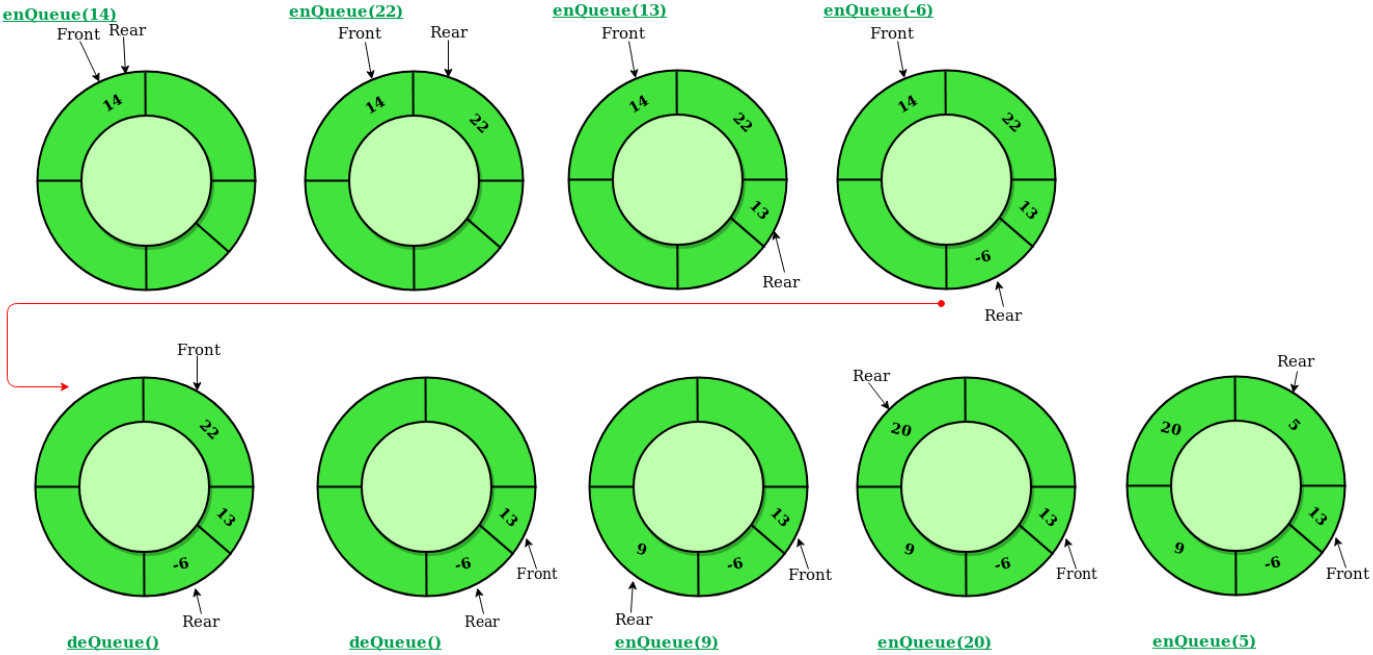






Passing the Ball





Note:

- For getting next rear value, the formula is **rear=(rear+1)%SIZE**
- For getting next front value, the formula is **front=(front+1) %SIZE**

Applications of Circular Queue :

- CPU scheduling
- Memory management
- Traffic Management

//Write a C program that implement Circular Queue and its operations using the arrays.

```
#include<stdio.h>
#include<stdlib.h>
#define QSIZE 5
int f=-1,r=-1;
void enqueue(int q[], int item)
{
    if ( ((f==0)&&(r==QSIZE-1)) || (f==r+1) )
        {
         printf("queue is full \n");
         return;
        }
   if((f==-1)&&(r==-1))
        f=r=0;
   else
        r=(r+1) % QSIZE;  // advance rear clockwise
   q[r]=item;
return;
}

void dequeue(int q[])
{
     int item;
     if((f==-1)&&(r==-1))
       {
        printf("queue is empty \n");
        return;
        }
     item=q[f];
     printf("the deleted item from queue is %d\n",item);
     if(f==r)
       f=r=-1;
     else
       f=(f+1)%QSIZE;  // advance front clockwise
```

```
return;
}

void display(int q[])
{
   int i;
   if((f==-1)&&(r==-1))
         {
          printf("queue is empty \n");
          return;
         }
   printf("\n elements in queue  ");
   for(i=f;i!=r;i=((i+1)%QSIZE))
          printf("%d \t ",q[i]);
   printf("%d \t ",q[i]);
return;
}

void main()
{
   int q[QSIZE],item,choice;
   for(;;)
      {
          printf(" \n 1. insert \n 2. delete \n 3. display \n 4. exit\n");
          printf("enter your choice \n");
          scanf("%d",&choice);
          switch(choice)
          {
          case 1: printf("enter price of the item \n");
                       scanf("%d",&item);
                  enqueue(q,item);
                  break;
          case 2: dequeue(q);break;
          case 3: display(q);break;
          case 4: exit(0); // break;
          default: printf(" invalid choice \n");
          }
     }
}
```