**Implementation of Deque using circular array:**

Deque or Double Ended Queue is a generalized version of Queue data structure that allows insert and delete at both ends.

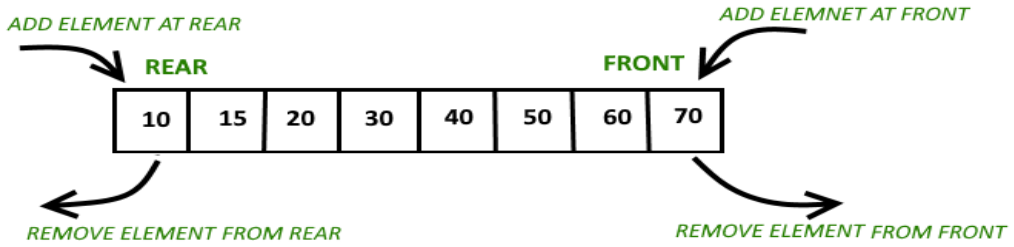
**Operations on Deque:**

Mainly the following four basic operations are performed on queue:

**insetFront()**: Adds an item at the front of Deque.
**insertRear()**: Adds an item at the rear of Deque.
**deleteFront()**: Deletes an item from front of Deque.
**deleteRear()**: Deletes an item from rear of Deque.
**display():**Displays elements in the Deque.

In addition to above operations, following operations are also supported

**getFront()**: Gets the front item from queue.
**getRear()**: Gets the last item from queue.
**isEmpty()**: Checks whether Deque is empty or not.
**isFull()**: Checks whether Deque is full or not.



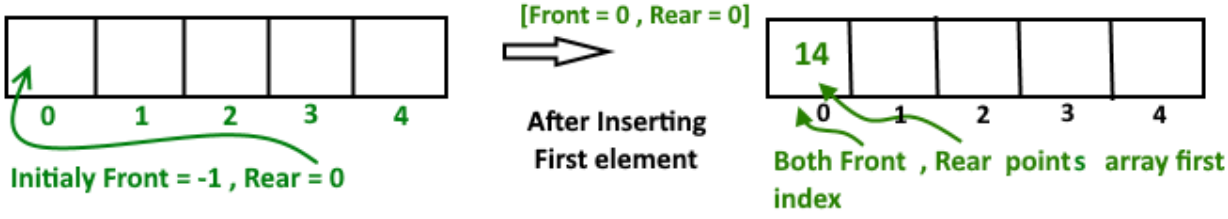

**Circular array implementation deque :**

For implementing deque, we need to keep track of two indices, front and rear. We enqueue(push) an item at the rear or the front end of deque and dequeue (pop) an item from both rear and front end.

**Working**

1. Create an empty array 'arr' of size 'n'
initialize **front = -1** , **rear = 0**

Inserting First element in deque, at either front or rear will lead to the same result.




After insert **Front** Points = 0 and **Rear** points = 0

**Insert Elements at Rear end**

a). First we check deque if Full or Not

b). IF Rear == Size-1

then reinitialize Rear = 0 ;

Else increment Rear by '1'

and push current key into Arr[ rear ] = key

Front remain same.

**Insert Elements at Front end**

a). First we check deque if Full or Not

b). IF Front == 0 || initial position, move Front

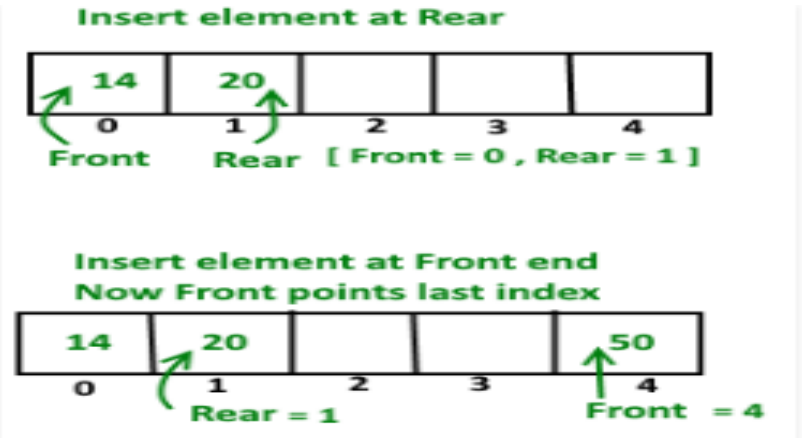
to points last index of array

front = size - 1

Else decremented front by '1' and push

current key into Arr[ Front] = key

Rear remain same.




**Delete Element From Rear end**

a). first Check deque is Empty or Not

b). If deque has only one element

front = -1 ; rear =-1 ;

Else IF Rear points to the first index of array

it's means we have to move rear to points

last index [ now first inserted element at

front end become rear end ]

rear = size-1 ;

Else || decrease rear by '1'

rear = rear-1;

**Delete Element From Front end**

a). first Check deque is Empty or Not

b). If deque has only one element

front = -1 ; rear =-1 ;

Else IF front points to the last index of the array

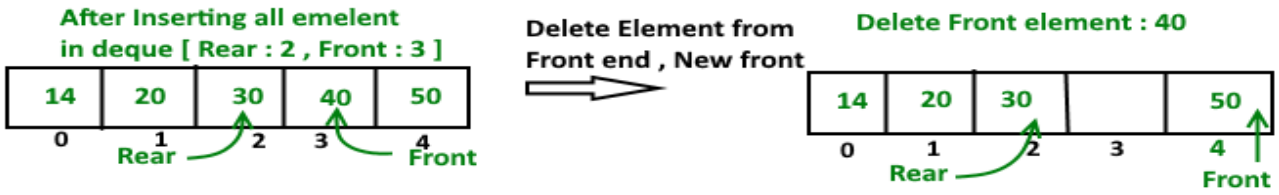
it's means we have no more elements in array so

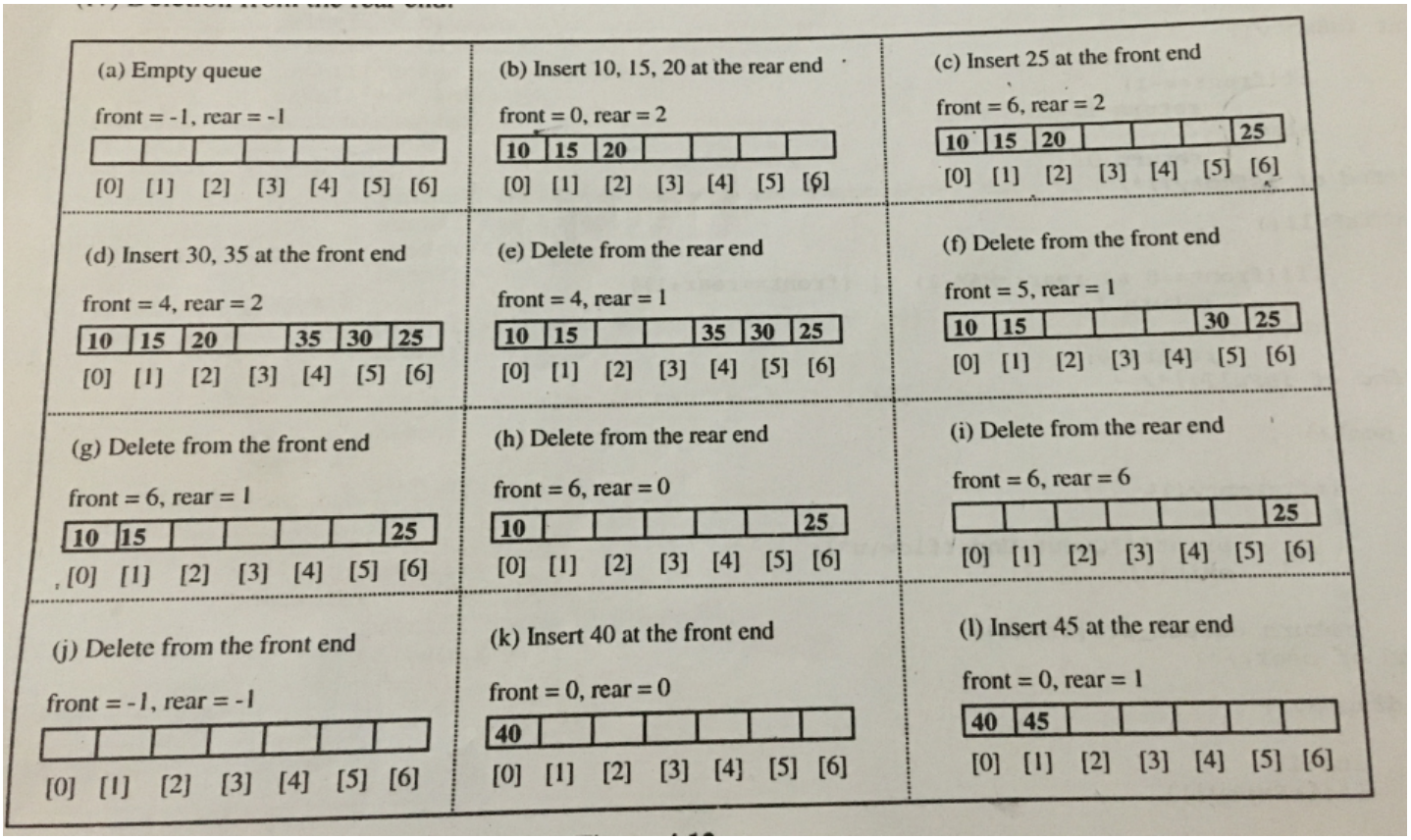
we move front to points first index of array

front = 0 ;

Else || increment Front by '1'

front = front+1;



**/*Program of deque using circular array*/**

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 7
int deque_arr[MAX];
int front=-1;
int rear=-1;
void insert_frontEnd(int item);
void insert_rearEnd(int item);
int delete_frontEnd();
int delete_rearEnd();
void display();
int isEmpty();
int isFull();
main()
{
        int choice,item;
        while(1)
        {
                printf("1.Insert at the front end\n");
                printf("2.Insert at the rear end\n");
                printf("3.Delete from front end\n");
                printf("4.Delete from rear end\n");
                printf("5.Display\n");
                printf("6.Quit\n");
                printf("Enter your choice : ");
                scanf("%d",&choice);
                switch(choice)
                {
                case 1:
                        printf("Input the element for adding in queue : ");
                        scanf("%d",&item);
                        insert_frontEnd(item);
                        break;
                case 2:
                        printf("Input the element for adding in queue : ");
                        scanf("%d",&item);
                        insert_rearEnd(item);
                        break;
                 case 3:
                        printf("Element deleted from front end is : %d\n",delete_frontEnd());
                        break;
                 case 4:
                        printf("Element deleted from rear end is : %d\n",delete_rearEnd());
                        break;
                 case 5:
```

```
                        display();
                        break;
                case 6:
                        exit(1);
                default:
                        printf("Wrong choice\n");
                }/*End of switch*/
                printf("front = %d, rear =%d\n", front , rear);
                display();
        }/*End of while*/
}/*End of main()*/

void insert_frontEnd(int item)
{
        if( isFull() )
        {
                printf("Queue Overflow\n");
                return;
        }
        if( front==-1 )/*If queue is initially empty*/
        {
                front=0;
                rear=0;
        }
        else if(front==0)
                front=MAX-1;
        else
                front=front-1;
        deque_arr[front]=item ;
}/*End of insert_frontEnd()*/

void insert_rearEnd(int item)
{
        if( isFull() )
        {
                printf("Queue Overflow\n");
                return;
        }
        if(front==-1)  /*if queue is initially empty*/
        {
                front=0;
                rear=0;
        }
        else if(rear==MAX-1)  /*rear is at last position of queue */
                rear=0;
        else
```

```
                rear=rear+1;
        deque_arr[rear]=item ;
}/*End of insert_rearEnd()*/

int delete_frontEnd()
{
        int item;
        if( isEmpty() )
        {
                printf("Queue Underflow\n");
                exit(1);
        }
        item=deque_arr[front];
        if(front==rear) /*Queue has only one element */
        {
                front=-1;
                rear=-1;
        }
        else
                if(front==MAX-1)
                        front=0;
                else
                        front=front+1;
        return item;
}/*End of delete_frontEnd()*/

int delete_rearEnd()
{
        int item;
        if( isEmpty() )
        {
                printf("Queue Underflow\n");
                exit(1);
        }
        item=deque_arr[rear];

        if(front==rear) /*queue has only one element*/
        {
                front=-1;
                rear=-1;
        }
        else if(rear==0)
                rear=MAX-1;
        else
                rear=rear-1;
        return item;
```

```
}/*End of delete_rearEnd() */

int isFull()
{
	if ( (front==0 && rear==MAX-1) || (front==rear+1) )
		return 1;
	else
		return 0;
}/*End of isFull()*/

int isEmpty()
{
	if( front == -1)
		return 1;
	else
		return 0;
}/*End of isEmpty()*/

void display()
{
	int i;
	if( isEmpty() )
	{
		printf("Queue is empty\n");
		return;
	}
	printf("Queue elements :\n");
	i=front;
	if( front<=rear )
	{
		while(i<=rear)
			printf("%d ",deque_arr[i++]);
	}
	else
	{
		while(i<=MAX-1)
			printf("%d ",deque_arr[i++]);
		i=0;
		while(i<=rear)
			printf("%d ",deque_arr[i++]);
	}
	printf("\n");
}/*End of display() */
```

**OUTPUT:**

```
1.Insert at the front end
2.Insert at the rear end
3.Delete from front end
4.Delete from rear end
5.Display
6.Quit
Enter your choice : 1
Input the element for adding in queue : 10
front = 0, rear =0
Queue elements :
10
1.Insert at the front end
2.Insert at the rear end
3.Delete from front end
4.Delete from rear end
5.Display
6.Quit
Enter your choice : 2
Input the element for adding in queue :
20
front = 0, rear =1
Queue elements :
10 20
1.Insert at the front end
2.Insert at the rear end
3.Delete from front end
```

```
4.Delete from rear end

5.Display

6.Quit


Enter your choice : 3

Element deleted from front end is : 10

front = 1, rear =1

Queue elements :

20

1.Insert at the front end

2.Insert at the rear end

3.Delete from front end

4.Delete from rear end

5.Display

6.Quit

Enter your choice : 5

Queue elements :

20

front = 1, rear =1

Queue elements :

20

1.Insert at the front end

2.Insert at the rear end

3.Delete from front end

4.Delete from rear end

5.Display

6.Quit
```

```
Enter your choice : 4

Element deleted from rear end is : 20

front = -1, rear =-
1

Queue is empty
```