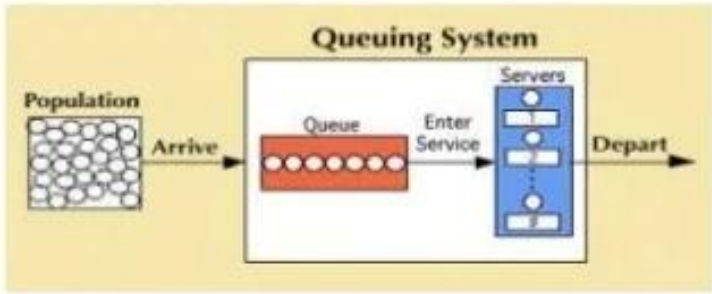
KONSEP QUEUE

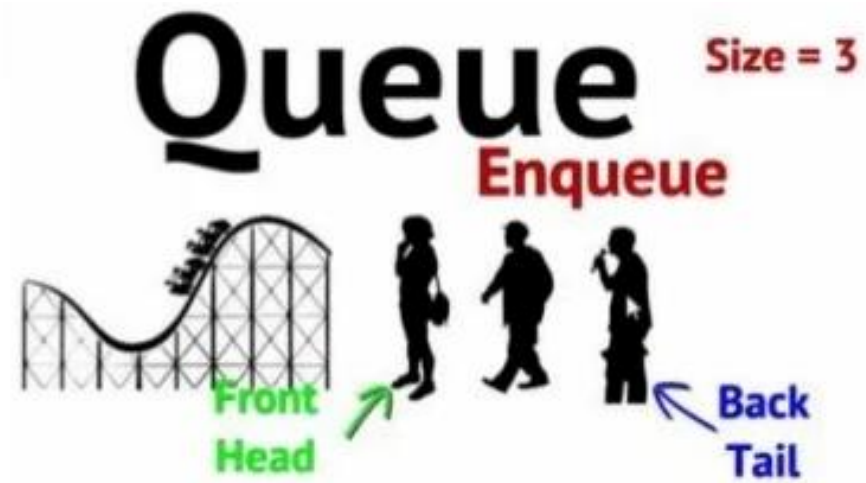
QUEUE

1. Queue atau antrian merupakan struktur data dengan operasi pemasukan data hanya diperbolehkan pada salah satu sisi, yang disebut sisi Belakang / ekor (Tail/Rear/back) dan operasi penghapusan hanya diperbolehkan pada sisi lainnya yang disebut sisi Depan / kepala (Head/Front).
2. Prinsip Queue adalah struktur data First In First Out (FIFO)

   First Come First Serve (FCFS)

QUEUE

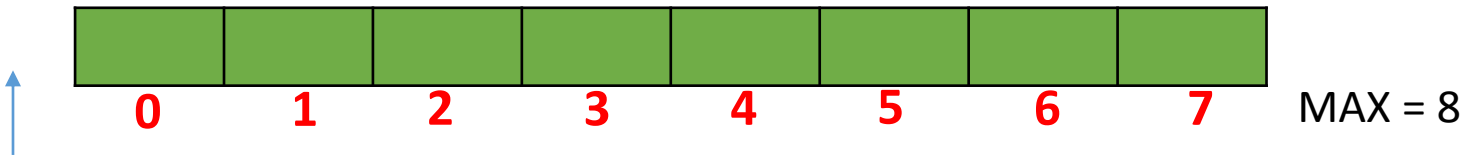

Queuing System
Population
Arrive
Queue
Enter
Service
Servers
Depart


Queue
Size = 3
Enqueue
Front
Head
Back
Tail

DEKLARASI QUEUE

Contoh kode:

```
#define max 8
typedef struct{
        int data[MAX];
        int head;
        int tail;
}Queue
Queue antrian;
```



head = -1

tail = -1

QUEUE

- Implementasi Queue:

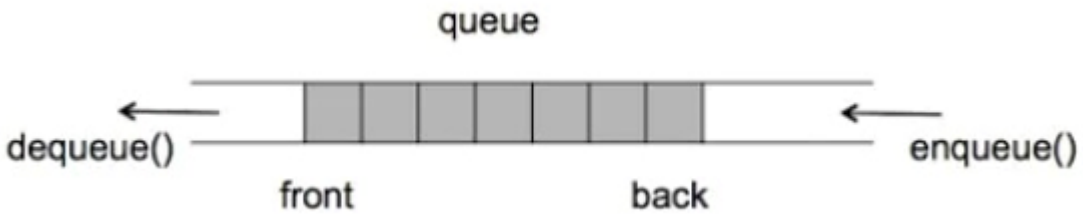
  a. Array (**statis**, ukuran queue diberikan diawal)

  b. Linked list (**dinamis**, tidak pernah penuh kecuali memorinya overflow)

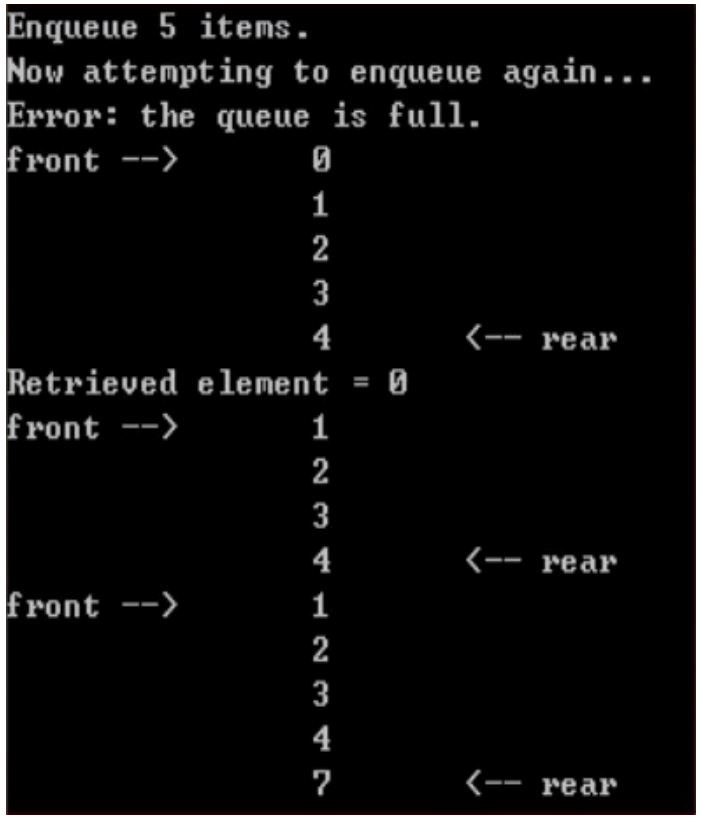
- Operasi dasar pada Queue:

 a. Push (Enqueue)

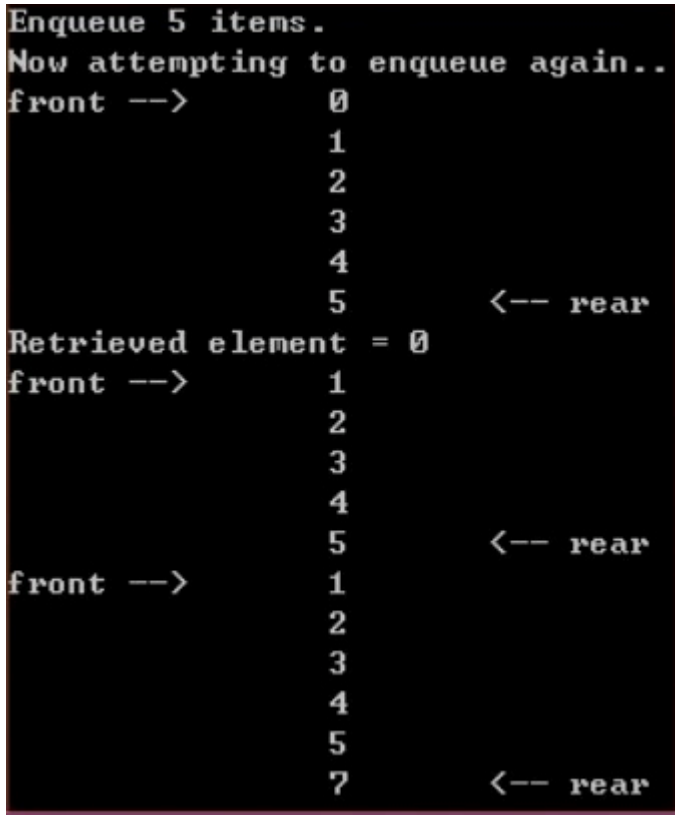
 b. Pop (Dequeue)



QUEUE



Array



Linked list

Queue yang diimplementasikan menggunakan Linked list tidak akan pernah full

OPERASI QUEUE

- **CREATE**

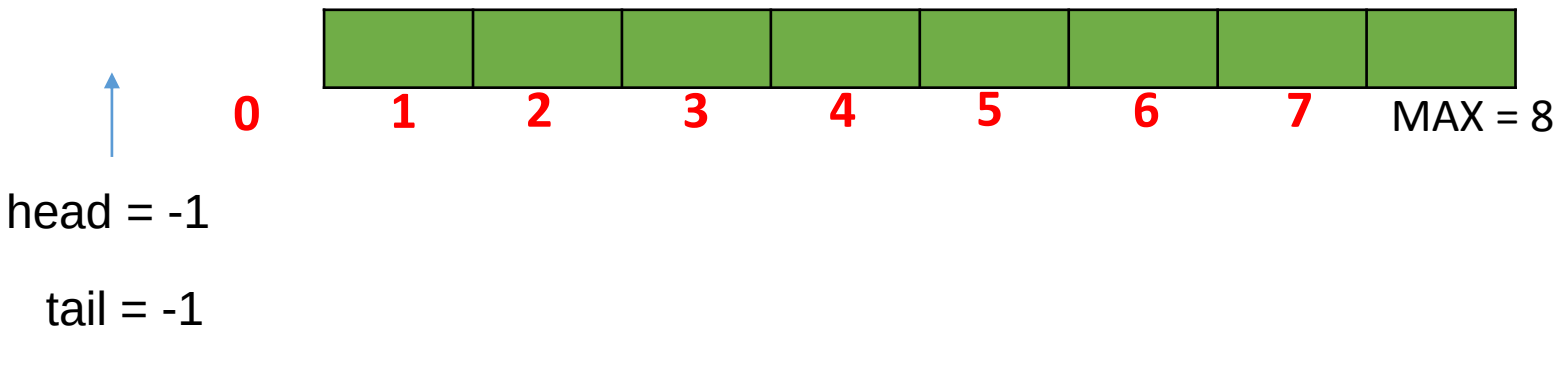
  Berfungsi untuk menciptakan dan menginisialisasi queue dengan cara membuat Head dan Tail = -1
- **ISEMPTY**
  Berfungsi untuk memeriksa apakah queue kosong.
- **ISFULL**
  Berfungsi untuk memeriksa apakah queue sudah penuh.
- **ENQUEUE**
  Berfungsi untuk menambahkan item pada posisi paling belakang.
- **DEQUEUE**
  Berfungsi untuk menghapus item dari posisi paling depan.
- **CLEAR**
  Untuk mengosongkan queue.

CREATE

Berfungsi untuk membentuk dan menunjukan awal terbentuknya suatu Queue.

Contoh kode:

```
Void Create()
{
    antrian.head = antrian.tail = -1
}
```



ISEMPTY

- Berfungsi untuk memeriksa queue penuh atau kosong.
- Penggunaan IsEmpty dengan cara memeriksa nilai Tail, jika Tail = -1 maka antrian kosong (empty).
- Head merupakan elemen pertama untuk kepala queue yang tidak akan berubah-ubah.
- Pergerakan queue terjadi dengan penambahan elemen queue ke belakang yaitu menggunakan nilai Tail

Contoh kode:

```
Int IsEmpty()
{
if (antrian.tail == -1)
        return 1;
else
        return 0;
}
```

ISFULL

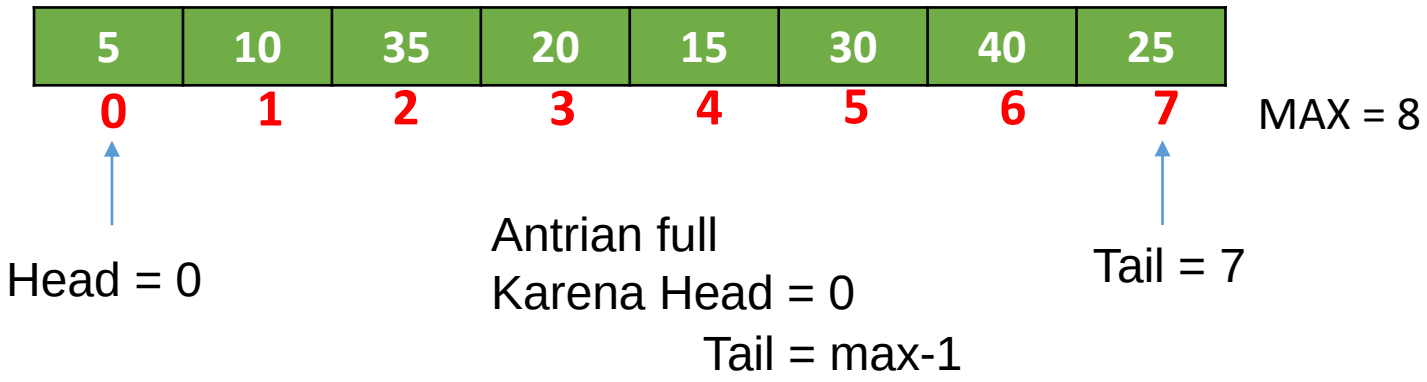
- Berfungsi untuk mengecek apakah Antrian sudah penuh atau belum
- Cara pengecekan, yaitu:

  a. Mengecek nilai Tail

  b. Jika tail = MAX-1 berarti antrian sudah penuh (MAX-1 adalah batas elemen array dalam program C++)

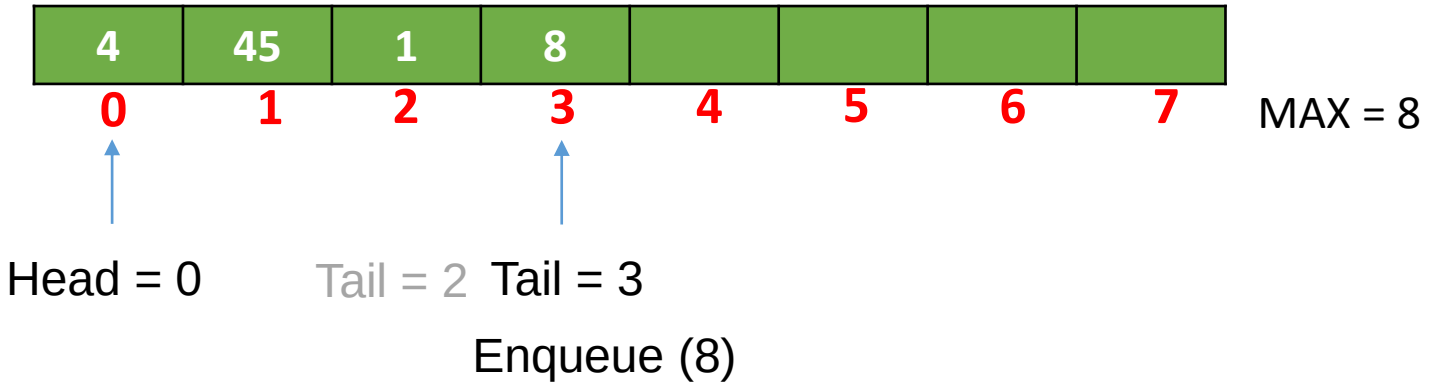
Contoh kode:

```
Int IsFull()
{
if (antrian.tail == Max-1)
    return 1;
 else
    return 0;
}
```



ENQUEUE

- Berfungsi untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu dilakukan pada elemen paling **belakang**
- Penambahan elemen selalu menggerakkan variabel Tail dengan cara menambahkan Tail terlebih dahulu.



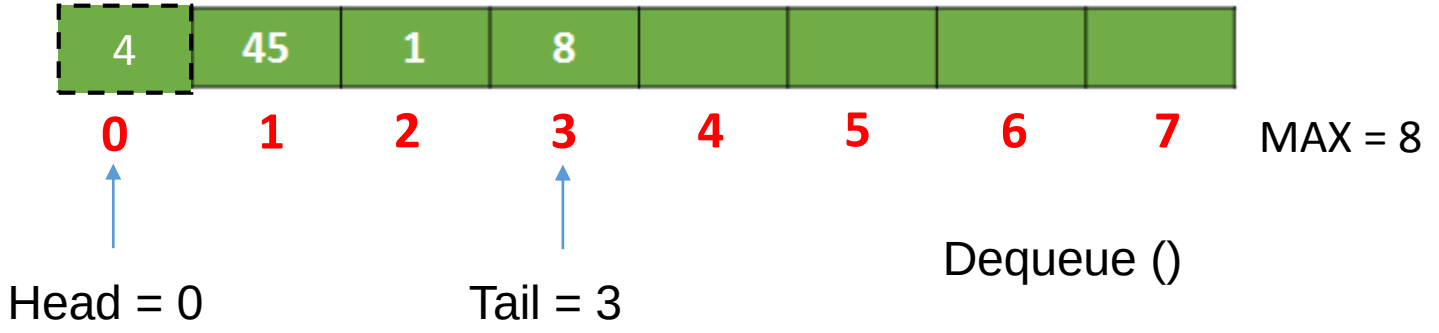
ENQUEUE

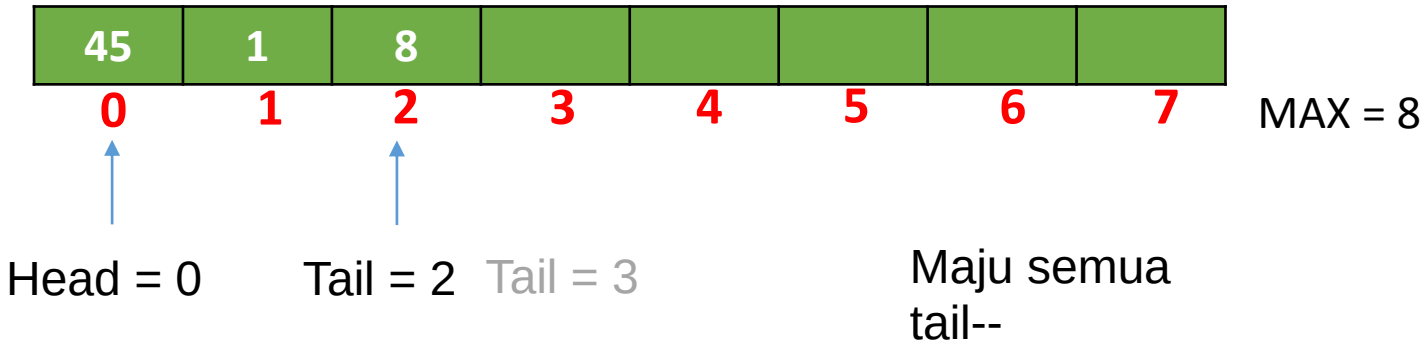
- Contoh kode:

```
void Enqueue(int data){
   if(IsEmpty()==1){
       antrian.head=antrian.tail=0;
       antrian.data(antrian.tail)=data;
       printf(%d masuk!”,antrian.data(antrian.tail)};
   }else
       if(IsFull()==0){
       antrian.tail++;
       antrian.data(antrian.tail)=data;
       printf(%d masuk!”,antrian.data(antrian.tail)};
```

DEQUEUE

- Berfungsi untuk menghapus elemen terdepan (head) dari Antrian
- Dengan cara: menggeser semua elemen antrian kedepan dan mengurangi Tail dgn 1. Penggeseran dilakukan dengan menggunakan looping





DEQUEUE

- Contoh kode:

```
int Dequeue(){
   int i;
   int e = antrian.data(antrian.head);
   for(i=antrian.head;i<=antrian.tail-1;i++){
       antrian.data(i) = antrian.data(i+1);
   }
   antrian.tail--;
   return e;
   }
```

CLEAR

- Berfungsi Untuk menghapus elemen-elemen Antrian dengan cara membuat Tail dan Head = -1
- Penghapusan elemen-elemen Antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesan- nya ke nilai -1 sehingga elemen-elemen Antrian tidak lagi terbaca sehingga mengembalikan antrian seperti keadaan semula

Contoh kode:

```
void Clear()
{
    antrian.head = antrian.tail = -1;
    printf("data clear");
}
```

JENIS QUEUE

1. Linier Queue
2. Circular Queue
3. Priority Queue
4. Deque (Double Ended Queue)

LINIER QUEUE

- Linier queue merupakan struktur data linier yang terdiri dari list pada setiap item.
- Pembahasan sebelumnya pada operasi queue menggunakan linier queue.
- Contoh:

Rear = 4 and Front = 1 and N = 7

10	50	30	40			
1	2	3	4	5	6	7

(1) Insert 20. Now Rear = 5 and Front = 1

10	50	30	40	20		
1	2	3	4	5	6	7

(2) Delete Front Element. Now Rear = 5 and Front = 2

	50	30	40	20		
1	2	3	4	5	6	7

(3) Delete Front Element. Now Rear = 5 and Front = 3

		30	40	20		
1	2	3	4	5	6	7

(4) Insert 60. Now Rear = 6 and Front = 3

		30	40	20	60	
1	2	3	4	5	6	7

LINIER QUEUE

Kelemahan linier Queue

setelah antrian penuh, meskipun beberapa elemen dari depan dihapus dan beberapa ruang yang ditempati dibebaskan, tidak mungkin untuk menambahkan elemen baru lagi, karena bagian belakang telah mencapai posisi paling belakang antrian

CIRCULAR QUEUE

- Queue ini tidak linier tetapi circular.
- Circular queue diaplikasikan untuk mengatasi kelemahan pada linier queue.
- Pada circular queue, setelah queue penuh, elemen pertama dari queue menjadi rear, jika dan hanya jika front telah bergerak maju. Jika tidak, itu akan menjadi status overflow queue.

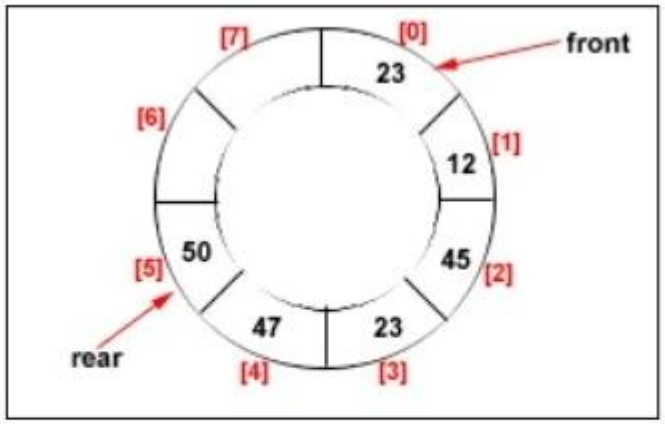
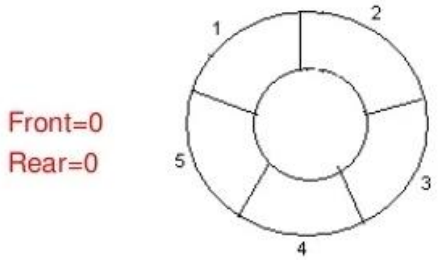


Figure: Circular Queue having Rear = 5 and Front = 0

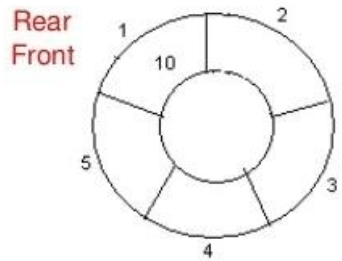
CIRCULAR QUEUE

Contoh: Circular Queue dengan N=5.

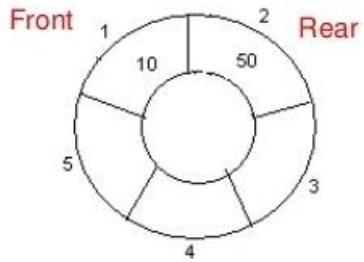
1. Initially, Rear = 0, Front = 0.



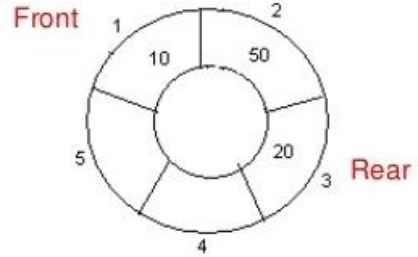
2. Insert 10, Rear = 1, Front = 1.



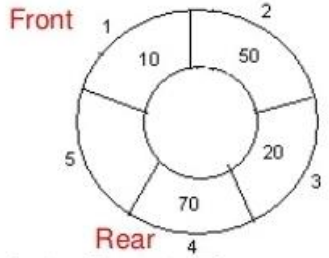
3. Insert 50, Rear = 2, Front = 1.



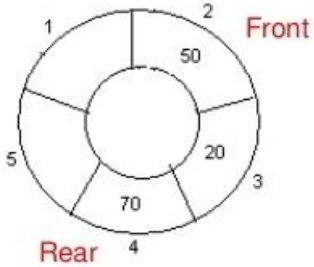
4. Insert 20, Rear = 3, Front = 0.



5. Insert 70, Rear = 4, Front = 1.

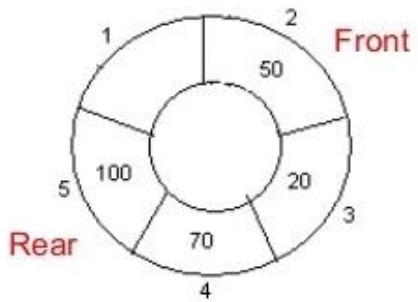


6. Delete front, Rear = 4, Front = 2.

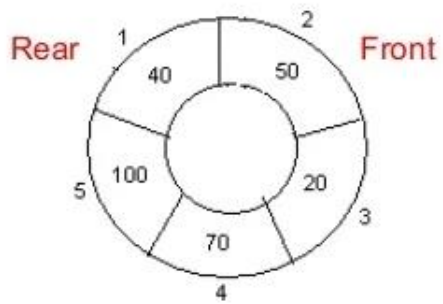


CIRCULAR QUEUE

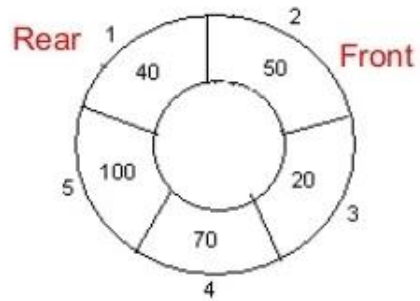
7. Insert 100, Rear = 5, Front = 2.



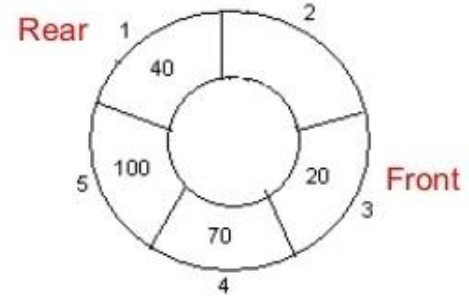
8. Insert 40, Rear = 1, Front = 2.



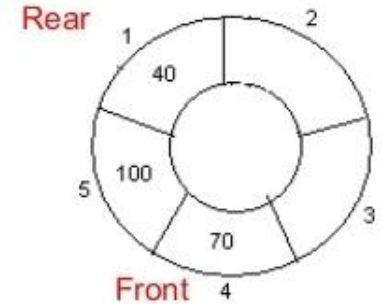
9. Insert 40, Rear = 1, Front = 2.
As Front = Rear + 1, so Queue overflow.



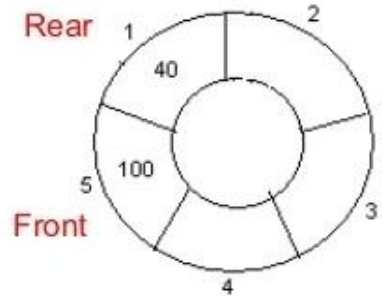
10. Delete front, Rear = 1, Front = 3.



11. Delete front, Rear = 1, Front = 4.



12. Delete front, Rear = 1, Front = 5.



PRIORITY QUEUE

- Priority queue merupakan special queue yang menambahkan atau menghapus elemen dilakukan sesuai dengan tingkat prioritasnya.
- Priority queue tidak menggunakan prinsip FIFO.
- Elemen dengan prioritas tinggi diproses terlebih dahulu dibandingkan dengan prioritas rendah.
- Karenanya prioritas tinggi di front queue.
- Priority queue dapat diimplementasikan pada:

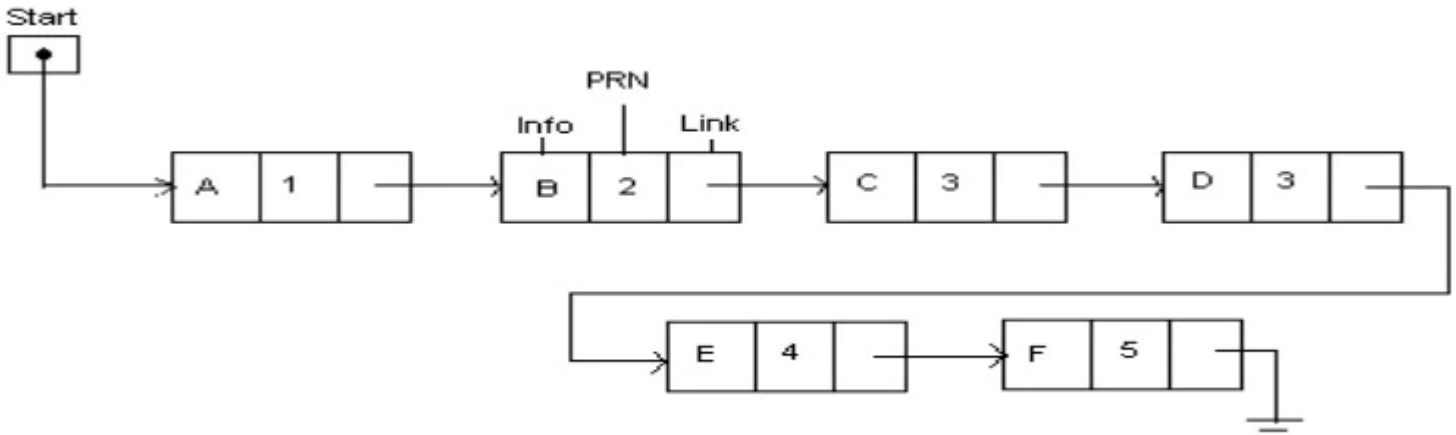
  a. Array(Representasi sequential)

  b. Linked list (Representasi Dynamic)

PRIORITY QUEUE

- Node priority queue terbagi menjadi tiga bagian, sesuai pada gambar berikut

Structure of priority queue Node

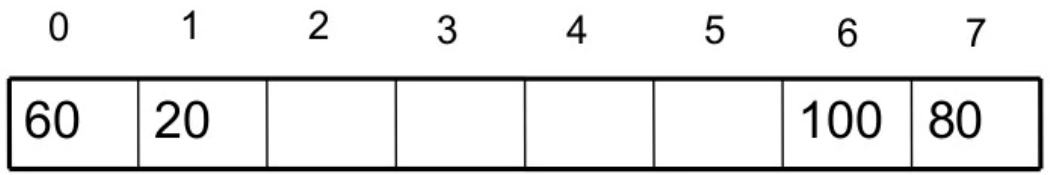


DEQUE

- Deque (double ended queue) merupakan list linier yang data elemennya dapat ditambahkan dan dihapus pada kedua ujungnya tetapi tidak bisa di tengah.
- Tipe deque:

a. **Input restricted deque**: penambahan diijinkan hanya satu ujung list tetapi penghapusan pada kedua ujung list

b. **Output restricted deque**: penambahan diijinkan pada kedua ujung list dan penghapusan hanya satu ujung list



TERIMA KASIH