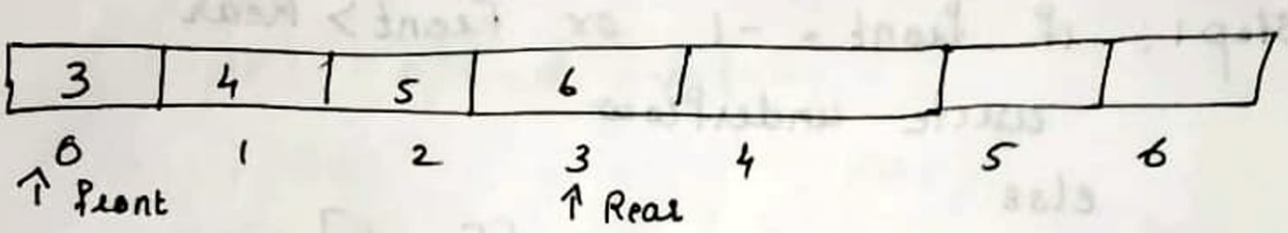
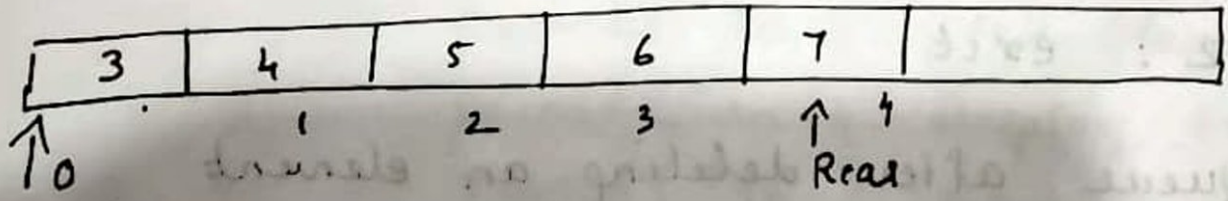
Queue

* Queue can be easily represented using linear Array:
* Two variables that point to the position from where insertion and deletion can be done such as Rear and Front respectively.



Front = 0, Rear = 3, Suppose we want to add ~~elem~~ element then Rear incremented by 1 So Rear = Rear + 1



Queue after insertion Front = 0 Rear = 4

Procedure: Enqueue

```
Step 1 : If Rear = = MAX - 1
             write overflow Go to step 4
         end If
       2 : If Front = -1 and Rear = = -1
             set front = Rear = 0
             else
                 Rear = Rear + 1
             end If
```

Step 3 : Set Queue[Rear] = Num

4 : exit.

Deleting [Dequeue]

1. check underflow condition, where try to delete an element from a queue that is already empty.

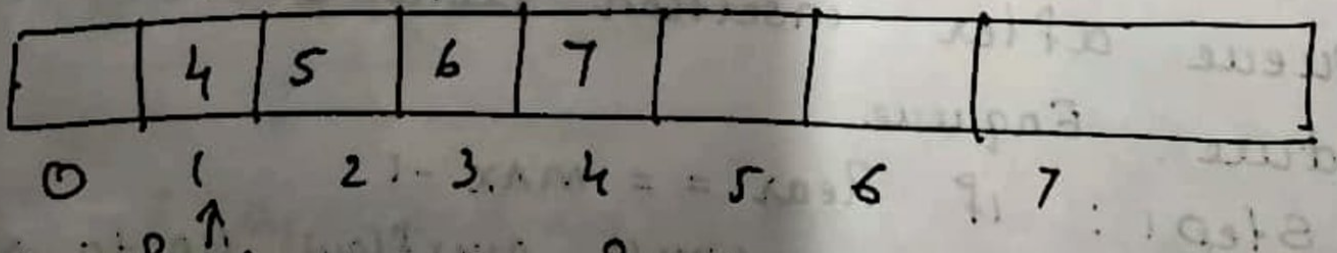
2. Front = -1 and Rear = -1, means no element in the Queue.

Algo: Dequeue

Step 1: if Front = -1 or Front > Rear
write underflow
else
set val = Queue[Front]
set Front = Front + 1
[end IF]

step 2 : exit

Queue after deleting an element



Front = 1 Rear = 4.

linear queue using array

```
#include <stdio.h>
#include <conio.h>
#define N 10
int queue[N];
int front = -1, rear = -1;
void insert (void);
int delete_element (void);
int peek(void);
void display (void);
int main ()
  {
    int option, val;
      do
        {
          printf("\n\n ** main menu ");
          printf("\n 1. insert an element");
          printf("\n 2. Delete an element");
          printf("\n 3. peek (top in queue/stack)");
          printf("\n 4. Display queue");
          printf("\n 5. Exit");
          printf("\n Enter option: ");
          Scanf ("%d", &option);
          Switch (option)
          {
           case 1 :
                    insert ();
                    break;
```

```
Case 2 :
        val = delete_element ();
        IF (val != -1 )
        PrintF ("\n the number deleted is :
                %d", val);
        break;

Case 3 :
        val = peek ()
        IF (val != -1 )
         PrintF ("\n The first value in
                Queue is : %d", val);
         break;

Case 4 :
        display ();
        break;
    }
} while (option != 5 );
   getch ();
   return 0;

}
```

```
void insert ()
{
  int num;
  printf ("\n Enter the number to be inserted in
           the queue : ");
  scanf ("%d", &num);
  if (rear == N-1)
     printf ("\n overflow");
  else if (front == -1 && rear == -1)
     front = rear = 0;
  else
     rear ++;
   Queue [rear] = num;
}
int delete_element ()
    {
      int val;
      if (front == -1 || front > rear)
        {
          printf ("\n underflow");
          return -1;
        }
      else
          {
            val = Queue [front];
            front ++;
            if (front > rear)
               front = rear = -1;
          }
          return (val
```

```
int peek ()
{
 if ( front == -1 || front > rear )
 {
  printf ("\n Queue empty");
    return -1;
  }
 else
   { return queue [front];
   }
}
void display ()
 {
   int i
   printf ("\n");
 if ( front == -1 || front > rear)
  printf ("\n Queue empty");
  else
       {
         for ( i = front; i <= rear; i++)
         printf ("\t %d", queue [i]);
         }
 }
```

O/P

** main menu **

1. insert
2. Delete
3. peek
4. display