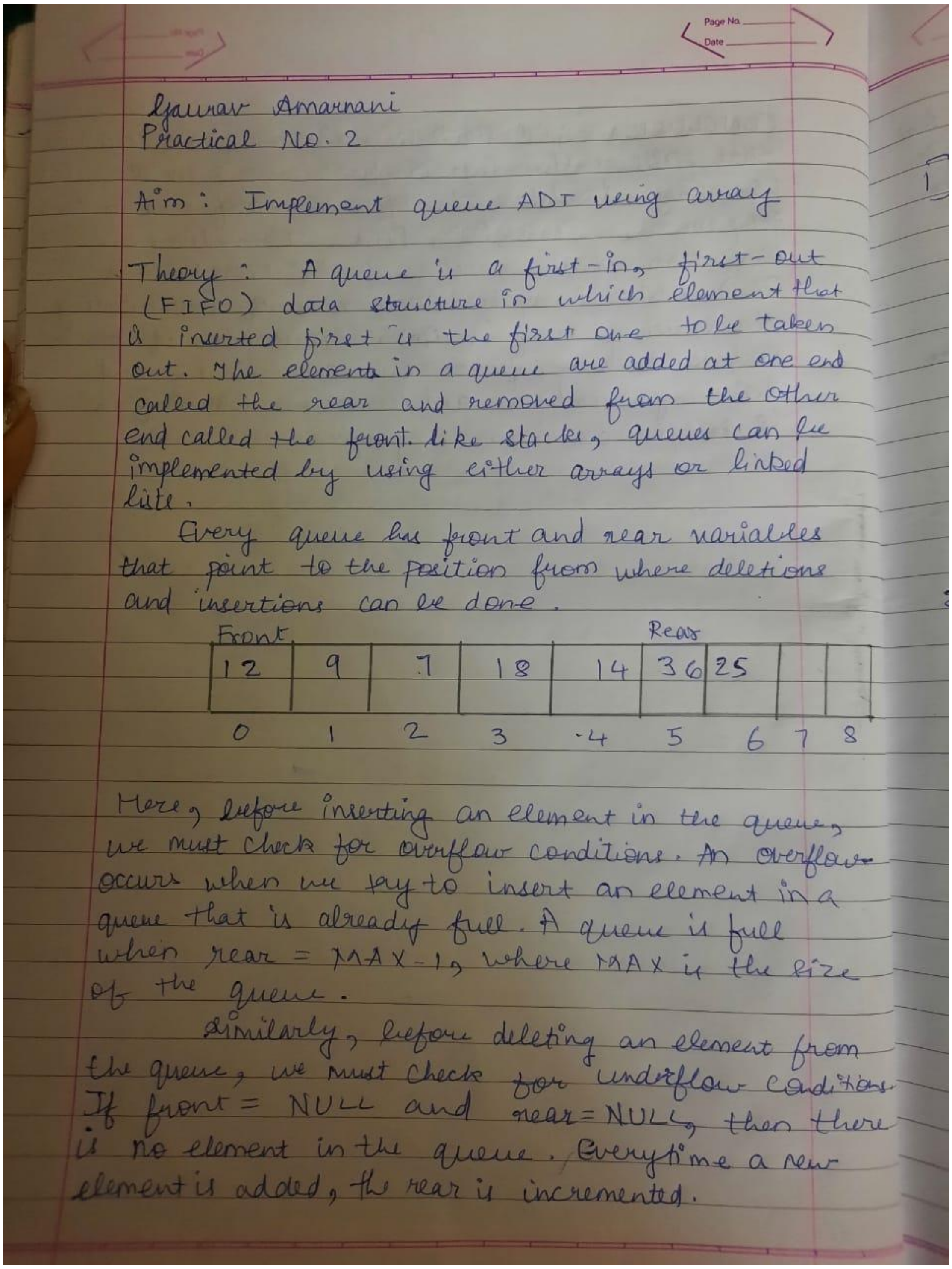
DS Assignment 2 - Gaurav Amarnani DSE CMPN.



Gaurav Amarnani
Practical No. 2

Aim : Implement queue ADT using array

Theory : A queue is a first-in, first-out (FIFO) data structure in which element that is inserted first is the first one to be taken out. The elements in a queue are added at one end called the rear and removed from the other end called the front. Like stacks, queues can be implemented by using either arrays or linked lists.

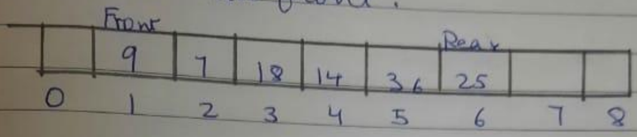
Every queue has front and rear variables that point to the position from where deletions and insertions can be done.

Front					Rear			
12	9	7	18	14	36	25		
0	1	2	3	4	5	6	7	8

Here, before inserting an element in the queue, we must check for overflow conditions. An overflow occurs when we try to insert an element in a queue that is already full. A queue is full when rear = MAX-1, where MAX is the size of the queue.

Similarly, before deleting an element from the queue, we must check for underflow conditions. If front = NULL and rear = NULL, then there is no element in the queue. Everytime a new element is added, the rear is incremented.

Now, if we want to delete an element from the queue, the value of front will be incremented. Deletions are done only from this end of the queue. Everytime an element is deleted, it is deleted from the front.



However, before inserting an element in the queue, we must check for overflow conditions. An overflow occurs when we try to insert an element into a queue that is already full. A queue is full when rear = MAX - 1, where MAX is the size of the queue, that is MAX specifies the maximum number of elements in the queue.

Similarly, before deleting an element from the queue, we must check for underflow conditions. If front = NULL and rear = NULL then there is no element in the queue.

Algorithm :

1] Enque (Inserting an element in Queue)

```
Step 1 : If REAR = MAX-1
                Write Overflow
                Go to step 4
Step 2 : If FRONT = -1 and REAR = -1
                Set FRONT = REAR = 0
         Else
                Set REAR = REAR + 1
Step 3 : SET QUEUE [REAR] = NUM
Step 4 : EXIT
```

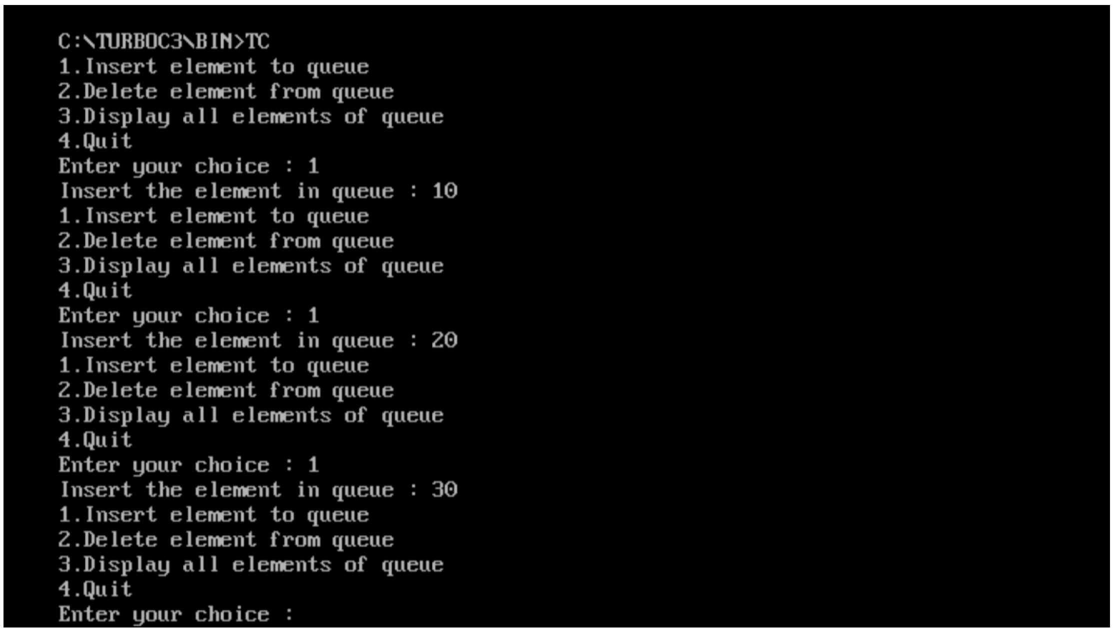
2] Deque (Deleting an element from a Queue)

```
Step 1 : If FRONT = -1 OR FRONT > REAR
                Write UNDERFLOW
         Else
                Set VAL = QUEUE [FRONT]
                Set FRONT = FRONT + 1
             [END OF If]
Step 2 : EXIT
```

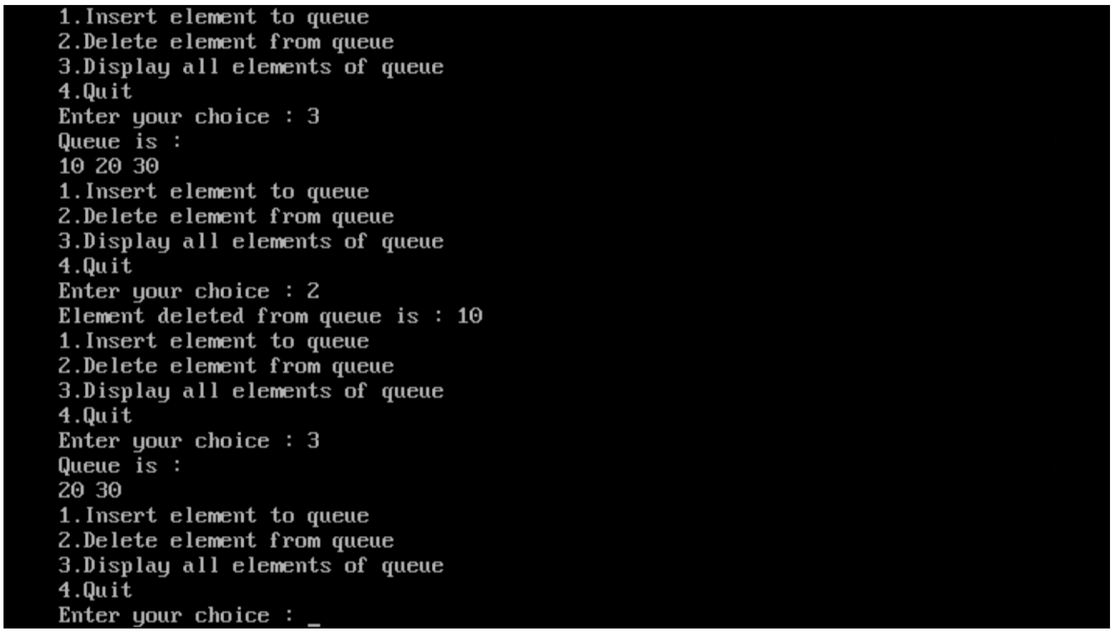
Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define MAX 50
void insert();
void delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
void main()
{
 int choice;
 while (1)
 {
 printf("1.Insert element to queue \n");
 printf("2.Delete element from queue \n");
 printf("3.Display all elements of queue \n");
 printf("4.Quit \n");
 printf("Enter your choice : ");
 scanf("%d", &choice);
 switch (choice)
 {
 case 1:
 insert();
 break;
 case 2:
 delete();
 break;
 case 3:
 display();
 break;
 default:
 printf("Wrong choice \n");
 break;
 }
 }
}
void insert()
{
 int add_item;
 if (rear == MAX - 1)
 printf("Queue Overflow \n");
 else
 {
 if (front == - 1)
 front = 0;
 printf("Insert the element in queue : ");
 scanf("%d", &add_item);
 rear = rear + 1;
 queue_array[rear] = add_item;
 }
}
void delete()
{
 if (front == - 1 || front > rear)
 {
 printf("Queue Underflow \n");
 return ;
 }
 else
 {
 printf("Element deleted from queue is : %d\n",
queue_array[front]);
 front = front + 1;
 }
}
void display()
{
 int i;
 if (front == - 1)
 printf("Queue is empty \n");
 else
 {
 printf("Queue is : \n");
 for (i = front; i <= rear; i++)
 printf("%d ", queue_array[i]);
 printf("\n");
 }
}
```

Output:

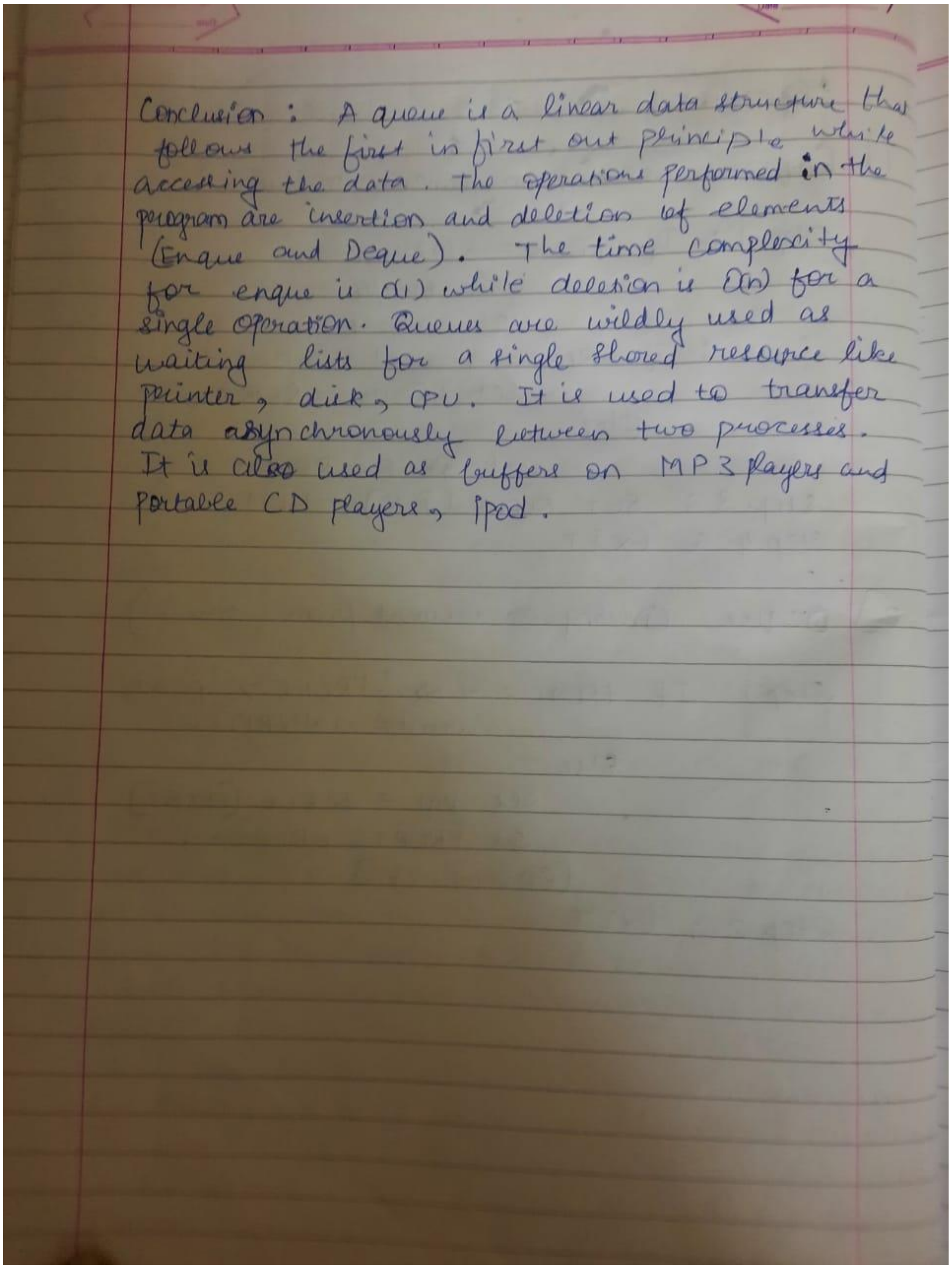


```
C:\TURBOC3\BIN>TC
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Insert the element in queue : 10
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Insert the element in queue : 20
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Insert the element in queue : 30
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice :
```



```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
10 20 30
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from queue is : 10
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
20 30
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : _
```

Conclusion:



Conclusion : A queue is a linear data structure that follows the first in first out principle while accessing the data. The operations performed in the program are insertion and deletion of elements (Enque and Deque). The time complexity for enque is O(1) while deletion is O(n) for a single operation. Queues are wildly used as waiting lists for a single shored resource like printer, disk, CPU. It is used to transfer data asynchronously between two processes. It is also used as buffers on MP3 players and portable CD players, ipod.