EXPERIMENT NO 04

AIM: Write a program to perform various operations on Queue.

SCOPE: Insertion and Deletion are the basic data structure operations. Through this experiment we will come to know that how insertion and deletion operation will be performed on Queue data structure.

FACILITIES: Software Needed: Turbo C

THEORY:

Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

A real-world example of queue can be a single-lane one-way road, where the vehicle enters first, exits first. More real-world examples can be seen as queues at the ticket windows and bus-stops.

Queue Representation

As we now understand that in queue, we access both ends for different reasons. The following diagram given below tries to explain queue representation as data structure −

Queue

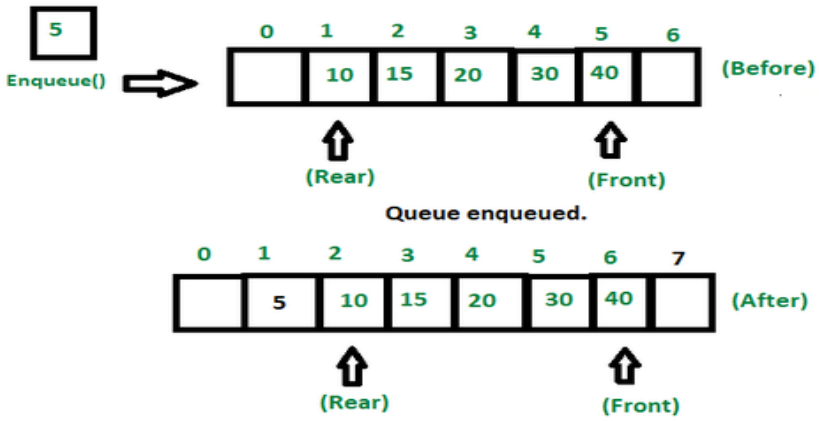
As in stacks, a queue can also be implemented using Arrays, Linked-lists, Pointers and Structures. For the sake of simplicity, we shall implement queues using one-dimensional array.

Operation: enqueue()

Inserts an element at the end of the queue i.e. at the rear end.

The following steps should be taken to enqueue (insert) data into a queue:

- Check if the queue is full.
- If the queue is full, return overflow error and exit.
- If the queue is not full, increment the rear pointer to point to the next empty space.
- Add the data element to the queue location, where the rear is pointing.
- return success.

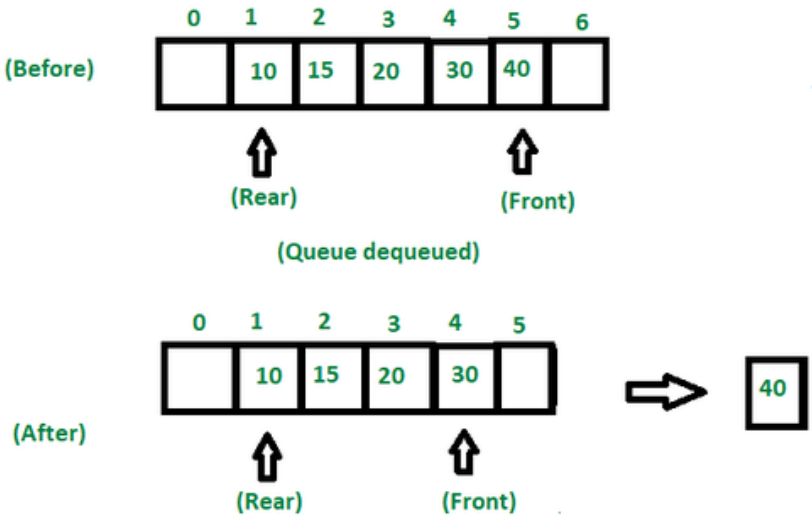


Operation: dequeue()

This operation removes and returns an element that is at the front end of the queue.

The following steps are taken to perform the dequeue operation:

- Check if the queue is empty.
- If the queue is empty, return the underflow error and exit.
- If the queue is not empty, access the data where the front is pointing.
- Increment the front pointer to point to the next available data element.
- The Return success.



IMPLEMENTATION:

//Program to implement various operations on queue

```
#include <stdio.h>
# define SIZE 100
void enqueue();
void dequeue();
void show();
int arr[SIZE];
int Rear = - 1;
int Front = - 1;
int main()
{
   int ch;
   while (1)
   {
      printf("1.Enqueue Operation\n");
      printf("2.Dequeue Operation\n");
      printf("3.Display the Queue\n");
      printf("4.Exit\n");
      printf("Enter your choice of operations : ");
      scanf("%d", &ch);
      switch (ch)
      {
         case 1:
         enqueue();
         break;
         case 2:
         dequeue();
         break;
         case 3:
         show();
         break;
         case 4:
         return 0;
         default:
         printf("Incorrect choice \n");
      }
   }
}
void enqueue()
{
   int insert_item;
   if (Rear == SIZE - 1)
      printf("Overflow \n");
   else
   {
      if (Front == - 1)
```

```
        Front = 0;
        printf("Element to be inserted in the Queue\n : ");
        scanf("%d", &insert_item);
        Rear = Rear + 1;
        arr[Rear] = insert_item;
    }
}

void dequeue()
{
    if (Front == - 1 )
    {
        printf("Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from the Queue: %d\n", arr[Front]);
        Front = Front + 1;
    }
}

void show()
{

    if (Front == - 1)
        printf("Empty Queue \n");
    else
    {
        printf("Queue: \n");
        for (int i = Front; i <= Rear; i++)
            printf("%d ", arr[i]);
        printf("\n");
    }
}
```

OUTPUT:

```
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Element to be inserted in the Queue: 10

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
```

Enter your choice of operations : 1
Element to be inserted in the Queue: 20

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 3
Queue:
10 20

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 2
Element deleted from the Queue: 10

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations: 3
Queue:
20

RESULT: In this way we have implemented different operations on queue.