ASSIGNMENT - 3

Name : B. Ravi Kumar

Reg NO : 192311246

Course : Data structure for stack overflow.

Course Code : CSA0389.

① Illustrate the que operation using following function calls of size = 5. Enque (25), Enqueue (37), Enqueue (90), Dequeue (), Enqueue (15), Enqueue (46), Enqueue (12), Dequeue(), Dequeue ().

Sol: To illustrate the queue operations for a queue of size 5 with the given sequence of function calls, Let's through each step.

Initial queue state:

* The Queue is empty initially
* maximum size of Queue is 5.

Operations:

1. Enqueue (25):

* Queue : '[25]'
* Front = 0, Rear = 0.

2. Enqueue (37):

* Queue : '[25, 37]'
* Front = 0, Rear = 1.

3. Enqueue (90):

* Queue : [25, 37, 90]
* Front = 0, Rear = 2.

4. Dequeue ():

* 25 is removed from the Queue.
* Queue : [37, 90]
* Front = 1, Rear = 2.

5. Enqueue (15):

* Queue : [37, 90, 15]
* Front = 1, Rear = 3

6. Enqueue (40):

* Queue : [37, 90, 15, 40]
* Front = 1, Rear = 4.

7. Enque (12):

* Queue = [37, 90, 15, 40, 12]

8. Dequeue ():

* 37 is removed from the Queue.
* Queue : [90, 15, 40, 12]

9. Dequeue ():

* 90 is removed from the Queue.
* Queue : '[15, 40, 12]'
* Front = 3, Rear = 5.

10. Dequeue ():

* 15 is removed from the queue.
* Queue : '[40, 12]'.

11. Dequeue ():

* 40 is removed from the queue.
* Queue : '[12]'.

Final Queue state :

* the queue contains '[12]' after all operations are performed.
* Front = 5, Rear = 5.

Summary of operation :

→ the operations performed show how elements are enqueued and dequeued from the Queue.

→ the queue's maximum size is never exceed, and elements are dequed in the order they were enqueued, following the First-in-first-out [FIFO].

(2) write a C program to implement queue operations such as ENQUEUE, DEQUEUE, and DISPLAY.

```
#include <stdio.h>
#include <stdlib.h>
#define size 5.

struct Queue {
        int .items [size] ;
        int front ;
        int rear ;
};
struct Queue * create Queue () {
    struct Queue * queue = (struct Queue *) malloc (size of (struct
                                Queue)) ;
    queue → front = -1 ;
    queue → rear = -1 ;
     return queue ; }.
int is full (struct Queue * queue) {
    if (queue → rear == size -1).
       return 1 ;
    return 0 ;
}.
int is Empty (struct Queue* queue) {.
   if ( queue → front == -1 || queue → front > queue→rear)
return 1 ;
}
```

```
void enqueue (struct Queue* queue , int value) {
 if (isFull (queue)){
    Printf ("Queue is full ! cannot enqueue %d\n", value);
}else {
    if (queue -> front == -1)
      queue -> front =0 ;
      queue -> rear ++ ;
      queue -> items [queue -> rear] = value ;
      Print f ("Enqueued %d \n", value) ;
    }
}
void    deque (struct Queue * queue) {
    if (isEmpty (queue)) {
    Printf ("Queue is Empty ! Cant dequeue \n");
    } else {
       queue -> front ++ ;
    }
}
int main () {
    Struct Queue* Queue = Create Queue ();
    enque (queue , 10) ;
    enqueue (queue, 20) ;
```

```
    enqueue (queue, 30);
    enqueue (queue, 40);
    enqueue (queue, 50);
    display (queue);
    display (queue);
    display (queue);
    enqueue (queue, 60);
    display (queue);
    dequeue (queue);
    dequeue (queue);
    display (queue);
    return 0;
}
```