Assignment - 4

Name : G. Harshitha

Reg No : 192324250

Course : Data structure

Course Code : CSA0389

Department : CSE(AI& DS)

9/10

Illustrate the queue operation using following funct
Calls of size : 5 Enqueue (25), Enqueue(37), Enque
Dequeue(), Enqueue(15), Enqueue (40), Enqueue(12), De
Dequeue(), Dequeue(), Dequeue()

To illustrate the queue operation for a queue 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 the queue : 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. Enqueue(12):

- * Queue : '[37, 90, 15, 40, 12]'
- * Front = 1, Rear = 5

8. Dequeue():

- * 37 is removed from the queue
- * Queue : '[90, 15, 40, 12]'
- * Front = 2, Rear = 5

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]'
- * Front = 4, Rear = 5.

Final Queue State:

* The queue contains '[12]' after all operations are performed

* Front = 5; Rear = 5

Summary of Operations:

=> The operations performed show how elements are equeued and dequeued from the queue

=> The queue's maximum size is never exceed, and elements are dequeued in the order they were enqueued, following the first - In - First - out [FIFO] principle.

② Write a C program to implement Queue operations Such as ENqueue, DeQueue and Display.

```
#include <stdio.h>
#include <stdlib.h>
#define size 5
Struck Queue{
        int . items [size];
        int . front;
        int . rear;
};
struck 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)
                return1;
        return0;
}
int is Empty (struct Queue* queue){
        if (queue -> front == -1 ||queue->front->re
return 0;

}
Void enqueue (struct Queue*queue, int value){
    if (is full (queue)){
        Printf ("Queue is full! Cannot enqueu%d \n", value);

    } else{
        if (queue-> front == -1)
                queue-> front = 0;
        queu-> rear++;
        queue-> items [queue -> rear]= value;
        printf ("Enqueued %d \n", value);

    }
}
Void dequeue (struct queue* queue) {
    if (isempty(queue)){
            printf( "queue is empty! Cannot dequeue
    }else{
        printf ("Dequed %d \n", queue->items [qu
        queue -> front ++;
    }
}
void display (struct queue* queue) {
        if (isempty (queue)){
        printf("Queue is empty \n");
    }
    Pointf("\n");
    }
}
int main() {
    struct Queue * queue = create Queue();
    enqueue(queue,10);
    enqueue (queue, 20);
    enqueue (queue, 30);
    enqueue (queue, 40);
    enqueue (queue,50);
    display (queue);
    display (queue)-
    display (queue);
    display (queue 60);
    display (queue);
    display (queue);
    display (queue);
    return 0;
}.
```

Output :-

```
Enqueued 10
Enqueued 20
Enqueued 30
Enqueued 40
Enqueued 50
Queue : 10 20 30 40 50
```

```
Deque 10
Queue: 20 30 40 50
Queue is full ! can not enque
queue : 20 30 40 50
Dequeued 20
Dequed 30
Queue : 40 50.
```