ASSIGNMENT-01

Name : A. Nagaveni

Reg no : 192311179

course : Data Structure

code : CSA0389

Date : 21 - 08 - 2024

I Illustrate the queue operation using following function calls of size =5. Enqueue (25), Enqueue(37), Enqueue (90), Dequeue(), Enqueue (15), Enqueue(60), Enqueue (12), Dequeue(), Dequeue(), 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 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,

11) Dequeue( ):

* 40 is removed from the queue.

Queue : '[12]'

* Front = 5, Rear = 5

Final Queue state: the queue contains [12] after all operations are performed.

* Front = 5, Rear = 5

Summary: => the operations performed show how elements are enqueued and dequeued from the queue.
=> the queue's max size is never exceed and elements are dequeued in the order they were enqueued, following the FIFO principal.

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 (sizeof(struct queue));
    queue -> front = -1;
    queue -> rear = -1;
    return queue;
}
int isfull (struct queue* queue) {
    if (queue -> rear == size - 1)
    return 0;
}
int isempty (struct queue* queue) {
    if (queue -> front == -1 || queue -> front -> queue -> rear)
    return 1;
    return 0;
}
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;
    }
}
void dequeue (struct queue* queue) {
    if (isEmpty (queue)) {
    printf ("Queue is Empty! cannot dequeue \n");
}
else {
    printf ("Dequeued %d \n", queue->items [queue->front]);
    queue -> front ++;
    }
    printf (" \n");
    }
}
int main() {
    struct queue* queue = createQueue();
    Enqueue (queue, 10);
    Enqueue (queue, 20);
    enqueue (queue, 30);
    Enqueue (queue, 40);
    Enqueue (queue, 50);
    display (queue);
    dequeue (queue);
    display (queue);
    Enqueue (queue, 60);
    display (queue);
    dequeue (queue);
    dequeue (queues);
    return 0;
}
```

output:

```
Enqueued 10
Enqueued 20
Enqueued 30
Enqueued 40
Dequeue 10
Queue : 20, 30, 40, 50
Queue is full; cannot Enqueue 60
Queue : 20, 30, 40, 50.
```