Name : Kvamsi Krishna yadav.

Reg no : 192324165.

Course : Data structure.

Course code: CSA0389

Submission : 21/08/2024

Assignment no: 5.

Instrate the queue operation using following function calls of size = 5. enqueue (25), enqueue(37), enqueue (90), enqueue(), enqueue (15), enqueue (40), enqueue (12), dequeue (), dequeue (), dequeue (), dequeue().

Sol To illustrate the queue operation 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.

operation:

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.
   * 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.

* front = 2, Rear = 5.

9. Dequeue ( ):

* 90 is removed from the queue.

* front = 3, Rear = 5.

10) Dequeue ( ):

* 15 is removed from the queue.

* front = 4, Rear = 5.

11) Dequeue ( ):

* 40 is removed from the queue.

* front = 5, Rear = 5.

Final queue state:

* The queue contains [12] after the all operations are Performed.

* front = 5, Rear = 5.

Summary of operation:

=> The operations Performed show how elements are enqueued and dequed from the queue.

=> The queue's max size is near exceed, and elements are dequed in the order they are. enqueued, following the first-In. first-out [fifo] Principle.

(2) write a C Program to implement queue operation such as enqueue, Dequeue and display.

Sol

```
#include <stdio.h>
#include <stdlib.h>
# define size 5
  struct queue {
        int items[size], front, 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;
       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;
            queue -> rear ++;
            queue -> item [queue -> rear]=value;
            Printf (" enqueued %d \n", value);
    }
}
void dequeue ( struct queue * queue) {
        * (isempty(queue){
        Printf ("queu is empty ! camot
                                    dequeue \n");
    } else {
        Printf ( "dequeued %d \n", queue->item[
                                    queue -> front]);
        queue -> front ++;
    }
};
```

```
void display (struct queue * queue){
    if (isEmpty (queue)){
        Printf ("queue: ");
        for (int i = queue->front; i<= queue->rear; i++){
        Printf ("%d", queue -> item[i]);
        Printf ("\n");
    }
}
int main(){
    struct queue * queue= createqueue();
    enqueue ( queue, 10);
    enqueue ( queue, 20);
    enqueue (queue, 30);
    enqueue (queue; 40);
    enqueue ( queue, 60);
    enqueue ( queue; 50);
    display (queue);
    display ( queue);
    display ( queue);
    display (queue);
    display ( queue);
    display (queue);
    return 0;
```

Output:

enqueued 10

enqueued 20

enqueued 30

enqueued 40

enqueued 50

enqueued 60

dequeue 10

queue : 20, 30 40 50

queue : is full cannot en

Dequeued : 20

Dequeued : 30

Dequeued: 40, 50, 60.

queue : 10, 20, 30, 40, 50, 60.