Assignment - 3

Name :- T. Sanhith

Reg no :- 192311228

COURSE code :- CSA0389

COURSE name :- Data structure

Date :- 21-Aug-2024

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

To illustrate the queue operations for a queue of size 5 with the given sequence of function calls, let's through each steps :-

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, 12]
   * 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 = 15

10. Dequeue() :

* 80 is removed from the queue
* Queue : '[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 of operations :-

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

=> The dequeued maximum size of never, exceed, and element are dequeued in the order they were enqueued following the First-in-first out [FIFO] principal.

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(){
struck queue* queue = (struck queue*) malloc(sizeof(
    struct Queue));
    Queue -> front = -1;
    Queue -> rear = -1;
     return queue;
}
  int is full (struck queue* queue){
    if (queue -> rear == size-1)
       return 1;
     return 0;
  }
    int is empty (struck 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;
      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\n");
  }else{
     printf("Dequeued %d\n", queue-> items[queue->
     front]);
       queue -> Front++;
     }
  }
   void display(struct Queue* queue){
       if (isempty(queue)){
         printf("Queue is empty !\n"); }else{
       else{
            printf("Queue:");
for(int i = queue -> front; i <= queue -> rear; i++){
     printf("%d ", queue -> items[i]);
   }
   printf("\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);
      dequeue (queue);
      display( queue);
      dequeue (queue);
      display( queue);
      dequeue(queue);
      dequeue(queue);
      display (queue);
      return 0; }
```

Output :-

```
Enqueued 10
enqueued 20
enqueued 30
enqueued 40
enqueued 50

Queue : 10 20 30 40 50

Dequeue 10

Queue : 20 30 40 50

Queue is full! cannot enqu
Dequeued 20
Dequeued 30

Queue : 40 50.
```