K. Nitheesh
192325090
CSA0369

Assignment - 3

1. Illustrate the queue operations using following function calls of size = 5. Enqueue (25), Enqueue (37), Enqueue (90), Dequeue(), Enqueue (15), Enqueue (40), Enqueue (12), 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 = 5
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 ( ) :-

- 90 is removed
- Queue : '[40, 12]'
- Front = 4, Rear = 5.

11. Dequeue :-

- 40 is removed
- 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 deque'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, & 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 0;
 }
```

```
}

void enqueue (struct Queue *queue, int value) {
    if (isfull (queue) {
        printf ("Queue is full! cannot Enque %d \n", value);
    } else {  if (queue -> front == -1)
                queue -> front = 0;
              queue -> rear++; }
}

int main () {
    struct Queue* Queue = createQueue ();
    Enqueue (queue, 10);
    Enque (queue, 20);
    Enqueue (queue, 30);
    enqueue (queue, 40);
    display (queue);
    display (queue);
    display (queue);
    return 0;
}
```

output :-

Enqueued 10
Enqueued 20
Enqueued 30
Enqueued 40
Enqueued 50

Deque : 10
Queue : 20 30 40 50
Queue is full ! cannot Enque 6
Queue : 20 30 40 50
Dequed : 20
Dequed : 30
Dequed : 40, 50

Queue : 10 20 30 40 50