

ASSIGNMENT - 04

Name : Egreeshma.v
Reg No : 192365012
Dept : CSE (cyber Security)
Course code : CSA0389
Course name : Data Structures
Faculty Name : Dr. Ashok Kumar
Assignment No : 04
Date of Submission : 21-Aug-2024.
No. of pages : 06.

1. Illustrate the queue operation using following function calls of size = 5. Enqueue(25), Enqueue(37), Enqueue(40), Dequeue(), Enqueue(15), Enqueue(40), 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: '[37]' '[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 of Operations:

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

⇒ The queue's maximum size is never exceeded, 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

Struct Queue {
    int items[SIZE];
    int front;
    int rear;
};

Struct Queue* createQueue() {
    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 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;
        queue->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 {
        printf ("Queue: ");
        for (int i = queue->front; i <= queue->rear; i++) {
            printf ("%d ", queue->items[i]);
        }
        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(queue);
    display(queue);
    return 0;
}
```

Output:

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

Dequeued 10

Queue: 20 30 40 50

Queue is full! cannot enqueue 60

Queue: 20 30 40 50

Dequeued 20

Dequeued 30

Queue: 40 50.