

## Assignment 03

NAME :- J. Lokesh

Reg no :- 190311216

SUB :- D.S

SUB CODE :- CSA0389.

### Assignment 3

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

To illustrate the queue operations for a queue of size 5 with the given sequence of functions 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

## Linear Queue

- \* The queue contains 'front' after all operations are performed
- \* Front is, Rear is

### Summary of operations:

- 1) The operations performed show elements are enqueued and dequeued from the queue.
- 2) The queue's maximum size is never exceed, and elements are dequeued in the order they were enqueued following the First-In-First-Out (FIFO) principle.

\*) Write a C program to implement Queue operation 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 QueueC() {
```

```
    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 (!is Full(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 (!is Empty(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 (!is Empty(queue)) {
```

```
printf("Queue is empty!\n");
```

```
} else {
```

```
printf("Queue: ");
```

```
for (int i = queue->front; i <= queue->rear; i++) {
```



```
printf("%d", queue[items]);
```

```
}  
printf("\n"); }
```

```
int main() {
```

```
struct Queue * queue = (struct Queue);
```

```
enqueue(queue, 10);
```

```
enqueue(queue, 20);
```

```
enqueue(queue, 30);
```

```
enqueue(queue, 40);
```

```
enqueue(queue, 50);
```

```
display(queue);
```

```
display(queue);
```

```
display(queue);
```

```
enqueue(queue, 60);
```

```
display(queue);
```

```
dequeue(queue);
```

```
dequeue(queue);
```

```
display(queue);
```

```
return;
```

```
}
```

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 enqueue

Queue: 20 30 40 50

Dequeue: 20

Dequeue: 30

Queue: 40 50