SIMATS

ASSIGNMENT NO- 3

COURSE CODE - CSA0389.

COURSE NAME - DATA STRUCTURE.

NAME - K. GOWTHAM REDDY.

REG NO - 192311220.

> 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.

Sol: Let's assume the queue has a size of 5.

<u>Initialise</u> <u>state</u> :

Queue : '[–, –, –, –, –]' (Empty)

front : –1

Rear : –1

1) Enqueue (25) :

Insert 25 at the rear

Queue after operation : [25 , –, –, –, –]

front : 0 [moved from –1 to 0]

Rear : 0 [moved from –1 to 0]

2) Enqueue(37) :

Insert 37 at the rear

Queue after operation : [25, 37 , –, –, –]

front : 0

rear : 1

3) Enqueue (90) :

Insert 37 at the rear

Queue after operation : [25 , 37 , 90, –, –]

front : 0 , Rear 2

4) Dequeue

Remove the element from the front

Queue after operation : [– , 37 , 90 , – , –]

front : 0

Rear : 2

5) ~~Deq~~ Enqueue (15)

~~Remove~~ Insert 15 at rear

Queue after operation : [– , 37 , 90 , 15 , –]

front : 1

rear : 3

6) Enqueue (40)

Insert 40 at rear

Queue after operation : [_ 37 , 90 , 15 , 40]

front : 1

Rear : 0

~~8~~ Dequeue ( )

Remove the element from the front

Queue after operation : [12 , – , 90 , 15 , 40]

front : 2

Rear : 0

9) Dequeue ( ) :

Remove the element from the front (i.e , 90)

Queue after operation [12 , – , – , 15 , 40]

front : 3

Rear : 0

i) Dequeue ()

Remove the element from the front

Queue after operation [12, -, -, -, 40]

front: 4

Rear: 0

ii) Dequeue():

Remove the element from the front.

queue after operation : [12, -, -, -, -]

front: 0

Rear: 0

Final state:

Queue : [12, -, -, -, -]

front : 0

Rear : 0

2) write a c program to implement Queue operations such as enqueue, Dequeue and display.

```
#include <stdio.h>
#define size 5;
struct Queue {
    int item [size];
    int front, rear;
};
void initialise (struct Queue *q ) {
    q-> front = q-> rear = -1;
}
```

```
int Is_Full (struct queue *q) {
      return (q->rear +1) % size == q->front ;
  }
int isEmpty (struct queue *q ) {
      return q->front == -1 ;
  }
void enque (struct queue *q ) {
    if (is empty (q)) { printf (" queue underflow" ) ; return ;
    int element = q->items [q - front] ;
    if (q-> front == q->rear ) q-> front = q->rear = -1 ;
    else {
       q->front = (q-> front +1 ) % size ;
       return element ; }
void display ( struct *q ) {
    int i = q-> front ;
    while (i <= q->rear ) {
       printf ("% d , q->item[i]) ;
         i = (i+1) % size ; }
  int main () {
    struct queue q ; initialize (&q ) ;
    enqueue (q , 25 ) ; enqueue (q , 37) ;
    dequeue (q) ;

    return 0;

  }
```