DATE:-

Assignment-3:

NAME:-G.Rahul

Regno:-192311244

DATA STRUCTURE

1: Illustrate the queue operation using following function calls of size = 5, Enqueue(25), Enqueue (37), Enqueue(90), Dequeue( ), Enqueue (15), Enqueue(49) Enqueue (12), Dequeue( ), Dequeue( ).

Let assume the queue has a Size of 5

Initalize state:-

* 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 '90' at the rear.
* Queue after operation [25, 37, 90, -, -]
* Front : 0 , * Rear : 1 .

④ Dequeue ( ):-

* Remove the elements from the Queue.
* Queue after operation [-, 37, 90, -, -]
* front : 1
* Rear : 2.

⑤ Enqueue(15):-

* Insert 15 at the 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 : 4.

⑦ Enqueue (12) :

* Insert '12' at the rear.
* Queue after operation [12,37,90,15,40]
* front : 1
* Rear : 4.

⑧ Dequeue() :

* Remove the element from the front (i.e 37)
* 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.

⑩ Dequeue() :

* Remove the element from the front (i.e, 15)
* Queue after operation [12,-,-,-,40]
* front : 4
* Rear : 0.

⑪ Dequeue() :

* Remove the element from the front (i.e, 40).
* 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 initalisize (Struct Queue *q) {
      q → front = q → rear = -1;
  }
  int isfull (struct Queue*q) {
    return (q → rear + 1) % Size = = q → front;
} int is Empty (Struct Queue *q) {
    return q → front == -1;
}
Void enqueue (struct Queue*q) {
 if (is empty (a) { 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 Queue *q) {
   if (isempty (q) printf ("Queue is Empty");
   else {
      int i = q → front;
    while (i ! = q → rear) {
        printf ("%d", q→item[i];
          }.
    Printf ("%d\n", q→items[q→rear]);
      }
  }
int main() {
    Struct Queue q; initialize ($q);
   enqueue ($q.25); enqueue ($q.37), enqueue
                                      ($q, 90);
   display (8.q); return 0;
  }.
```