To implement traffic monitoring system using queue

Name :- SUJAL NIMJE

Roll no :- 64

Subject :- DS, practical no : - 6

Code :-

Circular.h =>

```
#include <stdio.h>
#include <stdlib.h>
struct Queue  {
    int *arr;
    int front, rear, max;
};
int isFull(struct Queue * queue){
    return queue -> front == (queue -> rear + 1) % queue -> max;
}
int isEmpty(struct Queue * queue){
    return queue -> rear == -1;
}
void allocateMemory(struct Queue * queue, int max){
    queue -> arr = (int *)malloc(sizeof(int) * max);
    queue -> front = -1;
    queue -> rear = -1;
    queue -> max = max;
}
void enqueue(struct Queue * queue, int element){
    if(isFull(queue)){
        printf("\nQueue is full\n");
        return;
    }
    if(queue -> rear == -1){
       queue -> front = 0;
       queue -> rear = 0;
       queue -> arr[queue -> rear] = element;
       return;
    }
    queue -> rear = (queue -> rear + 1) % queue -> max;
    queue -> arr[queue -> rear] = element;
}
int dqueue(struct Queue * queue){

    if(isEmpty(queue)){
        printf("queue is Empty \n");
        return -1;
    }
    if(queue -> front == queue -> rear){
    int curr = queue -> arr[queue -> front];
        queue -> front = -1;
```

```
        queue -> rear = -1;
        return curr;
    }
    int curr = queue -> arr[queue -> front];
       queue -> front = (queue -> front + 1) % queue -> max;
       return curr;
}
void display (struct Queue * queue){
    int curr = queue -> front;

    if(isEmpty(queue)){
        printf("Queue is Empty\n");
        return;
    }
    while(curr != queue -> rear){
        printf("%d ", queue -> arr[curr]);
        curr = (curr + 1) % queue -> max;
    }
     if(queue -> rear == curr){
        printf("%d \n", queue -> arr[curr]);
    }
    printf("\n");
}
```

Mainfile

```
#include <stdio.h>
#include "circular.h"

void main()
{
   struct Queue queue[4];
   int i = 0, elem, lane;
   int choice = 1;
   while(choice)
   {
    printf("enter the choice : 1 for entering full lane || 2 for display\n|| 3
for removal of lane || 4 for special entry : ");
    scanf("%d", &choice);
    switch(choice){

    case 1 :
   while(i < 4)
   {
   printf("enter the max size of the queue : ");
   int size;
   scanf("%d", &size);

   allocateMemory(&queue[i], size + 2);

     while(size > 0)
     {
     printf("enter the element : ");
     scanf("%d", &elem);
```

```
     enqueue(&queue[i], elem);
          size--;
     }
     i++;
   }
   break;
   case 2:
    i = 0;
       while(i < 4)
       {
          display(&queue[i]);
          i++;
       }
   break;
   case 3:
   while(1){

   if(isEmpty(&queue[0]) && isEmpty(&queue[1]) && isEmpty(&queue[2])  &&
isEmpty(&queue[3])){
        printf("all lanes are empty\n");
        return;
   }
       i = 0;
       while(i < 4)
       {
       int count = 0;
       if(!(isEmpty(&queue[i]))){
            if ( i == 0){
             printf("these is frist the lane -> \n");
            }
            if ( i == 1){
             printf("these is second the lane -> \n");
            }
            if ( i == 2){
             printf("these is third the lane -> \n");
            }
            if ( i == 3){
             printf("these is last the lane -> \n");
            }
            printf("\n");
       }
          while(!(isEmpty(&queue[i])))
          {
          if(count >= 5)
          {
               break;
          }
          printf("removed element is :  %d\n", dqueue(&queue[i]));
          count++;
          }
         i++;
       }
   }
   break;
   case 4:
```

```
        printf("enter the special enqueud lane : ");
        scanf("%d", &lane);
        printf("enter the element: ");
        scanf("%d", &elem);
        enqueue(&queue[lane - 1], elem);
        break;

   default :
   printf("an invalid input\n");
}}}
```

Output

```
PS C:\Users\SUJAL NIMJE\Downloads\practical 6> gcc traffic.c
PS C:\Users\SUJAL NIMJE\Downloads\practical 6> ./a.exe
enter the choice : 1 for entering full lane || 2 for display
|| 3 for removal of lane || 4 for special entry : 1
enter the max size of the queue : 1
enter the element : 1111
enter the max size of the queue : 2
enter the element : 2221
enter the element : 2222
enter the max size of the queue : 3
enter the element : 3331
enter the element : 3332
enter the element : 3333
enter the max size of the queue : 6
enter the element : 6661
enter the element : 6662
enter the element : 6663
enter the element : 6664
enter the element : 6665
enter the element : 6666
enter the choice : 1 for entering full lane || 2 for display
|| 3 for removal of lane || 4 for special entry : 2
1111

2221 2222

3331 3332 3333

6661 6662 6663 6664 6665 6666

enter the choice : 1 for entering full lane || 2 for display
|| 3 for removal of lane || 4 for special entry : 4
enter the special enqueud lane : 1
enter the element: 1112
enter the choice : 1 for entering full lane || 2 for display
|| 3 for removal of lane || 4 for special entry : 2
1111 1112

2221 2222
```

```
3331 3332 3333

6661 6662 6663 6664 6665 6666

enter the choice : 1 for entering full lane || 2 for display
|| 3 for removal of lane || 4 for special entry : 3
these is frist the lane ->

removed element is :  1111
removed element is :  1112
these is second the lane ->

removed element is :  2221
removed element is :  2222
these is third the lane ->

removed element is :  3331
removed element is :  3332
removed element is :  3333
these is last the lane ->

removed element is :  6661
removed element is :  6662
removed element is :  6663
removed element is :  6664
removed element is :  6665
these is last the lane ->

removed element is :  6666
all lanes are empty
PS C:\Users\SUJAL NIMJE\Downloads\practical 6>
```