URBAN EDGE 11 / 1 / 24

LAB - 03 -

Circular Queue-

```
# include <stdio.h>
# define   SIZE   5

int   items [SIZE];
int   front  = -1  , rear = -1;

int   isFULL()  {
      if ((front == rear + 1) || (front == 0  && rear == SIZE -1))
      return 1;
      return 0;
}

int   isEmpty()  {
      if (front == -1)  return 1;
      return 0;
}

void  enQueue  (int elements) {
      if (isFull())
           printf ("\n Queue  is  full !!  \n");
      else {
         if  (front == -1 )  front = 0;
         rear  = (rear +1)  %  SIZE;
         items [rear] =  elements;
         printf ("\n  Inserted  ->  %d", element);
      }
}
```

```
int  deQueue() {
    int element;
    if (isEmpty()) {
        printf("\n Queue is empty !! \n");
        return (-1);
    }
    else {
        element = items[front];
        if (front == rear)
        {
            front = -1;
            rear = -1;
        }
    else
    {
        front = (front +1) % SIZE;
    }
    printf("\n . Deleted element -> %d \n", element);
    return (element);
  }
}

Void  display()
{
   int i;
   if (isEmpty())
       printf("\n Empty Queue \n");
   else
   {
       printf("\n Front -> %d", front);
       printf("\n Items -> ");
       for (i = front ; i != rear ; i = (i+1) % SIZE)
       {
           printf ("%d", items[i]);
       }
```

4.1.24

```
           printf("%d ", items[i]);
           printf("\n Rear -> %d \n", rear);
       }
    printf("\n");
}

Void  main()
{
   int option , val;
    int ele;
    do
    {
        printf(" 1. insert \n");
        printf(" 2. Delete \n");
        printf(" 3 Display \n");
        printf(" 4. Exit \n");
        printf(" enter your option : \n");
        scanf("%d", & option);
        switch (option)
        {
            case 1 :
                printf(" enter the element : ");
                scanf("%d" , & ele);
                enQueue (ele);
                break;
            case 2 :
                val = dequeue();
                if (val != -1)
                    printf(" the number deleted is :
                                   %d ", val);
                break;
            case 3 :
                display();
                break;
        }
    } while (option != 4);
}
```

Output

```
1. insert
2. Delete
3. Display
4. exit
enta your option : 1
1
enta the element : 1

    Inserted -> 1
1. insert
2. Delete
3. display
4 exit
enter your option :
1
enter the element : 2

    Inserted -> 2
1. Insert
2 delete
3. display
4 exit
enter your option :
2

    Deleted element -> 1
the number deleted is : 1 5
1. insert
2 delete
3. display
4 exit
```

11.1.24

```
enter your option :
3

Front -> 1
Items -> 2
Rear -> 1
```

11/1/24

Single Linked list

```
#include<stdio.h>
# include <stdlib.h>

Struct Node
{
    int data;
    Struct Node *next;
};

Void insert (Struct Node ** head, int data)
{
    Struct Node *newnode = (Struct Node *)malloc
              (size of (Struct node));

     newnode ->
     data = data;
     newnode ->

       * head = newnode;
}

Void display (Struct Node * node)
{
    printf ("\n Linked List : ");
    while (node != NULL)
    {
        printf ("%d ", node -> data);
        node = node -> next;
    }
    printf ("\n");
}
```

11/1/24

```
Void main()
{
    Struct Node * head = NULL;
    insert (& head, 95);
    insert (& head, 85);
    Insert (& head, 65);
    insert (& head, 45);
    insert (& head, 25);
    display (head);
}
```

output:-

```
Linked list : 95 85 65 45 25
```

[illegible]
11/1/24