Week 4

11/01/24 // 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 element)
{
    if (isFull())
        printf("\n Queue is full!! \n");
    else
    {
        if (front == -1)
            front = 0;
        rear = (rear + 1) % SIZE;
        items[rear] = element;
        printf("\n Inserted -> %d", element);
    }
    printf("\n");
}
```

```
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);
    }
    printf("\n");
}
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]);
        }
        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("read element:");
                scanf("%d", &ele);
                enQueue(ele);
                break;
            case 2:
                print
```

```
            val = dequeue();
            if (val != -1)
                printf("element is deleted is: %d\n"
                        , val);
            break;
        case 3:
            display();
            break;
        }
    } while (option != 4);
}
```

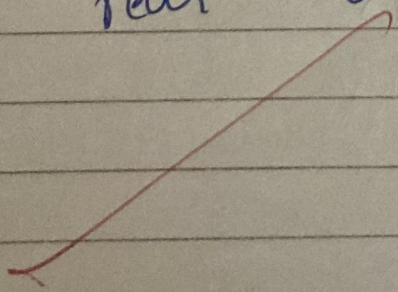
output:-

```
1. insert
2. Delete
3. Display
4. Exit
enter your option:
1
read the element: 45
    Inserted --> 45
1. Insert
2. Delete
3. display
4. Exit
enter your option: 2
    Deleted element --> 45
the element deleted is: 45
1. insert
2. Delete
3. Display
4. Exit
```

enter your option:
2

Queue is empty!!
1. insert
2. Delete
3. Display
4. Exit
enter your option: 4
1
·read the element: 3
Inserted → 3
1. insert
2. Delete
3. Display
4. Exit
enter your option:
3

Front → 0
Items → 3
rear → 0



// Singly 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
    (sizeof(struct Node));
    newnode->
    data=data;
    new node->
    next= *head
    *head= newnode;
}
void display(struct Node* node)
{
    printf("\n Linked list: ");
    while(node!=NULL)
    {
        printf("%d ", node->data);
        node= node->next;
    }
    printf("\n");
}
void main()
{
```

```
struct Node * head=NULL;
insert(&head, 80);
insert(&head, 100);
insert(&head, 60);
insert(&head, 40);
insert(&head, 20);
display(head);
```

output:-

Linked List: 20 40 60 80 100.

Sol:
10/1/24