Question:
Design and implement a given type of (ordinary queue, circular queue) queue in C (array implementation/ Linked list implementation). And demonstrate its working with suitable
inputs. Display appropriate messages in case of exceptions.

Aim:
To Implement Linear Queue using Linked Lists

Algorithm:

Enqueue:
- Create a new node.
- If the value of the node is NULL that means the system has run out of memory to allocate so throw an exception indicating Overflow and return the control flow
- Else make the value of the node the data that the user provided
- Also if ‘front’ points to NULL then make then point towards the newly created node.
- And also make their next value point to NULL
- If ‘front’ doesn't point to NULL then make rear’s next value point to the newly created node
- Now make rear point to the new node
- And set rear’s next to NULL

Dequeue:
- Firstly check if front is NULL
- If it is NULL then it means that the queue is empty and we run into the underflow situation, so throw an exception indicating underflow and return the control flow
- If front is not NULL then store the value that front is currently pointing to in a temporary variable and make front point to the next node
- Now return the value that was previously stored.

Display

- Make a new node pointer which points to head.
- If the pointer points to null then display a message saying that the stack is empty
- Or else print the value of the current and then make the pointer point towards the next node
- Repeat step 3 till the pointer points to null.

Program

```
#include <stdio.h>
    #include <stdlib.h>
    struct node{
        int val;
        struct node *next;
    };
    typedef struct node node;
    node *front, *rear;


    void enqueue(int ele){
        node *ptr;
        ptr=(node*)malloc(sizeof(node));
        if(ptr==NULL)
            printf("overflow");
        else{
            ptr->val=ele;
            if(front==NULL){
                front=ptr;
                rear=ptr;
                front->next=NULL;
                rear->next=NULL;
            }
            else{
                rear->next=ptr;
                rear=ptr;
                rear->next=NULL;
            }
        }
    }
```

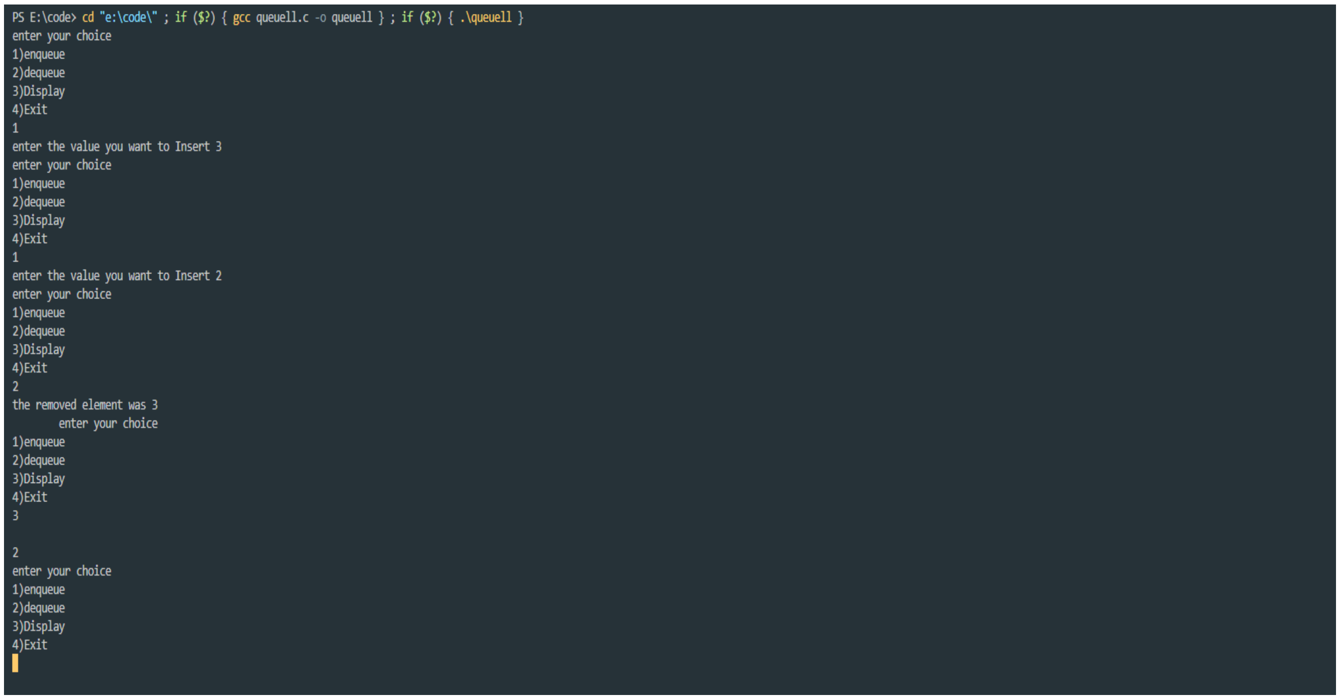
```
int dequeue(){
    node *ptr;
    if(front==NULL){
        printf("underflow");
        return -1;
     }
    else{
        int ele=front->val;
        ptr=front;
        front=front->next;
        free(front);
        return ele;
    }
    }

void display(){
    node *temp;
    temp=front;
    if(temp==NULL){
        printf("underflow");
        return;
     }
    else{
        while(temp!=NULL){
            printf("\n%d",temp->val);
            temp=temp->next;
            }
            printf("\n");
        }

}
int main(){
    int choice;
    while(1){
        printf("enter your choice \n1)enqueue \n2)dequeue
\n3)Display \n4)Exit \n");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                printf("enter the value you want to Insert ");
```

```
                    int ele;
                    scanf("%d",&ele);
                    enqueue(ele);
                    break;
                case 2:
                    printf("the removed element was %d\n
",dequeue());
                    break;
                case 3:
                    display();
                    break;
                case 4:
                    exit(1);
                    break;
                default:
                    printf("invalid");
            }
        }
    }
```

Output



```
PS E:\code> cd "e:\code\" ; if ($?) { gcc queuell.c -o queuell } ; if ($?) { .\queuell }
enter your choice
1)enqueue
2)dequeue
3)Display
4)Exit
1
enter the value you want to Insert 3
enter your choice
1)enqueue
2)dequeue
3)Display
4)Exit
1
enter the value you want to Insert 2
enter your choice
1)enqueue
2)dequeue
3)Display
4)Exit
2
the removed element was 3
        enter your choice
1)enqueue
2)dequeue
3)Display
4)Exit
3

2
enter your choice
1)enqueue
2)dequeue
3)Display
4)Exit
```