

**Aim:**

Write a program to implement queue using **linked lists**.

Sample Input and Output:

```
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 1
Enter element : 57
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 1
Enter element : 87
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 5
Queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 3
Elements in the queue : 57 87
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 2
Deleted value = 57
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 2
Deleted value = 87
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 3
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option : 6
```

**Source Code:**

QUsingLL.c

```
#include<stdio.h>
#include<conio.h>
void enqueue (int );
void dequeue();
void display();
void size();
void isEmpty();
int main() {
    int op,X;
    while(1) {
        printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
        printf("Enter your option : ");
        scanf("%d",&op);
        switch(op) {
```

```

        case 1:
        printf("Enter element : ");
        scanf("%d",&X);
        enqueue(X);
        break;
        case 2:
        dequeue();
        break;
        case 3:
        display();
        break;
        case 4:
        isEmpty();
        break;
        case 5:
        size();
        break;
        case 6: exit(0);
    }
}
}
struct queue {
    int data;
    struct queue *next;
};
typedef struct queue *Q;
Q front = NULL, rear = NULL;
void enqueue(int ele)
{
    Q temp = NULL;
    temp = (Q)malloc(sizeof(struct queue));
    if(temp == NULL)
    {
        printf("Queue is overflow.\n");
    }
    else {
        temp->data = ele;
        temp->next = NULL;
        if(front == NULL) {
            front = temp;
        }
        else {
            rear->next = temp;
        }
        rear = temp;
        printf("Successfully inserted.\n");
    }
}
void dequeue() {
    if(front == NULL)
        printf("Queue is underflow.\n");
    else {
        Q temp = front;
        front = front->next;
        printf("Deleted value = %d\n",temp->data);
        free(temp);
    }
}

```

```

    }
}
void display() {
    if(front == NULL)
        printf("Queue is empty.\n");
    else {
        Q temp = front;
        printf("Elements in the queue : ");
        while(temp!=NULL) {
            printf("%d ",temp->data);
            temp = temp->next;
        }
        printf("\n");
    }
}
void size() {
    int temp, count = 0;
    printf("Queue size : ");
    if(front == NULL)
        printf("0\n");
    else {
        Q temp = front;
        while(temp!=NULL) {
            count++;
            temp = temp->next;
        }
        printf("%d\n",count);
    }
}
void isEmpty() {
    if(front == NULL)
        printf("Queue is empty.\n");
    else
        printf("Queue is not empty.\n");
}
}

```

### Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option : 2
Queue is underflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option : 3
Queue is empty. 4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Enter your option : 4
Queue is empty. 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Enter your option : 5
Queue size : 0 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1

Enter your option : 1
Enter element : 44
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option : 1
Enter element : 55
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option : 1
Enter element : 66
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option : 1
Enter element : 67
Successfully inserted. 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option : 3
Elements in the queue : 44 55 66 67 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option : 2
Deleted value = 44 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option : 2
Deleted value = 55 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Enter your option : 5
Queue size : 2 4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Enter your option : 4
Queue is not empty. 6
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 6
Enter your option : 6

Test Case - 2
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option : 1
Enter element : 23
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option : 1
Enter element : 234
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option : 1
Enter element : 45
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option : 1
Enter element : 456
Successfully inserted. 2

1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option : 2
Deleted value = 23 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option : 3
Elements in the queue : 234 45 456 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option : 2
Deleted value = 234 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option : 3
Elements in the queue : 45 456 4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Enter your option : 4
Queue is not empty. 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Enter your option : 5
Queue size : 2 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option : 3
Elements in the queue : 45 456 6
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 6
Enter your option : 6