S.No: 6	Exp. Name: ***Queue using Linked Lists***	Date:2025-08-17

Aim:

Write a program that allows users to perform the following operations on a queue:

1. Enqueue an element (add to the rear).
2. Dequeue an element (remove from the front).
3. Display all elements in the queue.
4. Check if the queue is empty.
5. Get the size of the queue.
6. Exit the program.

Input Format:

The program displays a menu with the following options:

1. Enqueue
2. Dequeue
3. Display
4. Is Empty
5. Size
6. Exit

The user selects an option by entering a number corresponding to the desired operation.
For the Enqueue operation, the user is prompted to enter the integer element to be added to the queue.

Output Format:

For each operation, the program outputs the result:

1. For Enqueue, print: "Successfully inserted"
2. For Dequeue, print: "Deleted value: <X>" where <X> is the dequeued element, or "Queue is underflow" if the queue is empty.
3. For Display, print: "Elements: <element1><element2>....." showing all elements in the queue or "Queue is empty" if there are no elements.
4. For Is Empty, print: "Queue is empty" or "Queue is not empty".
5. For Size, print: "Queue size: <N>" where <N> is the number of elements in the queue. If the queue has no elements, print: "Queue size: 0"
6. For Exit, terminate the program without additional output.

Source Code:

QueueUsingLL.c

```
#include <stdlib.h>
#include <stdio.h>
#include "QueueOperationsLL.c"
int main() {
   int op, x;
   while(1) {
      printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
      printf("Option: ");
      scanf("%d",&op);
      switch(op) {
         case 1:
            printf("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);
      }
   }
}
```

QueueOperationsLL.c

```
//write your code here
struct Node{
   int data;
   struct Node* next;
};
struct Node* front = NULL;
struct Node* rear = NULL;

void enqueue(int x){
   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
   if(!temp){
      printf("Successfully inserted\n");
      return;
   }
   temp->data = x;
   temp->next = NULL;

   if(rear == NULL)
         front = rear = temp;
   else{
      rear->next = temp;
      rear = temp;
   }
   printf("Successfully inserted\n");
}

void dequeue(){
   if(front == NULL){
      printf("Queue is underflow\n");
      return;
   }
   struct Node* temp = front;
   printf("Deleted value: %d\n", front->data);
   front = front->next;
```

```
    if(front == NULL)
       rear = NULL;

    free(temp);

 }

 void display(){
    if(front == NULL){
       printf("Queue is empty\n");
       return;
    }
    struct Node* temp = front;
    printf("Elements: ");
    while(temp!=NULL){
       printf("%d ", temp->data);
       temp = temp->next;
    }
    printf("\n");
 }

 void isEmpty(){
    if(front == NULL)
       printf("Queue is empty\n");
    else
       printf("Queue is not empty\n");
 }

 void size(){
    int count = 0;
    struct Node* temp = front;
    while(temp!=NULL){
       count++;
       temp = temp->next;
    }
    printf("Queue size: %d\n", count);
 }
```

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
Option: 2
Queue is underflow 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Option: 3
Queue is empty 4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Option: 4
Queue is empty 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Option: 5

Queue size: 0 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Option: 1
element: 44
Successfully inserted 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Option: 1
element: 55
Successfully inserted 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Option: 1
element: 66
Successfully inserted 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Option: 1
element: 67
Successfully inserted 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Option: 3
Elements: 44 55 66 67 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Option: 2
Deleted value: 44 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Option: 2
Deleted value: 55 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Option: 5
Queue size: 2 4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Option: 4
Queue is not empty 6
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 6
Option: 6

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

Option: 1
element: 456
Successfully inserted 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Option: 2
Deleted value: 23 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Option: 3
Elements: 234 45 456 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Option: 2
Deleted value: 234 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Option: 3
Elements: 45 456 4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Option: 4
Queue is not empty 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Option: 5
Queue size: 2 6
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 6
Option: 6