Assignment 2: Queue Using Linked List

Objective:
To implement Queue data structure using a linked list in C.

Theory:
A Queue follows FIFO (First In First Out). Using a linked list allows dynamic memory allocation without size limitation.

Operations:

1. Enqueue: Insert element at the rear.
2. Dequeue: Remove element from the front.
3. Display: Show queue elements.

C Program:

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

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

struct Node *front = NULL, *rear = NULL;

void enqueue(int value) {
        struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = value;
        newNode->next = NULL;

        if (rear == NULL) {
        front = rear = newNode;
        return;
        }
        rear->next = newNode;
        rear = newNode;
```

```
}

void dequeue() {
        if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }
        struct Node *temp = front;
        front = front->next;
        free(temp);
}

void display() {
        struct Node *temp = front;
        while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
        }
        printf("\n");
}

int main() {
        enqueue(10);
        enqueue(20);
        enqueue(30);
        display();
        dequeue();
        display();
        return 0;
}
```

Input:

Enqueue: 10, 20, 30

Output:

```
10 20 30
 20 30
```

Conclusion:

Queue using linked list is efficient and flexible due to dynamic memory allocation.