

## Lab-06

WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.

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

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

// Insert at the beginning
void insertAtBeginning(struct Node** head_ref, int new_data) {
    // Allocate memory to a node
    struct Node* new_node
        = (struct Node*) malloc(sizeof(struct Node));

    // Put the data into the node
    new_node->data = new_data;

    // This new node is going to be the first node,
    // so set next of it as head
    new_node->next = (*head_ref);

    // Move the head to point to the new node
    (*head_ref) = new_node;
}

// Insert at the end
void insertAtEnd(struct Node** head_ref, int new_data) {
    // Allocate memory to a node
    struct Node* new_node
        = (struct Node*) malloc(sizeof(struct Node));

    // Put the data into the node
    new_node->data = new_data;

    // This new node is going to be the last node,
    // so set next of it as NULL
    new_node->next = NULL;

    if (*head_ref == NULL) {
        // If the Linked List is empty,
        // then make the new node as head
        *head_ref = new_node;
    }
}
```

```

    return;
}

// Else traverse till the last node
struct Node* last = *head_ref;
while (last->next != NULL)
    last = last->next;

// Change the next of the last node
last->next = new_node;
return;
}

// Display the linked list
void display(struct Node* head) {
    struct Node* ptr;
    ptr = head;
    while (ptr != NULL) {
        printf("%d -> ", ptr->data);
        ptr = ptr->next;
    }
    printf("NULL\n");
}

// Sort the linked list
void sortLinkedList(struct Node** head_ref) {
    struct Node* current = *head_ref;
    struct Node* index = NULL;
    int temp;

    if (head_ref == NULL) {
        return;
    } else {
        while (current != NULL) {
            // index points to the node next to current
            index = current->next;

            while (index != NULL) {
                if (current->data > index->data) {
                    temp = current->data;
                    current->data = index->data;
                    index->data = temp;
                }
                index = index->next;
            }
            current = current->next;
        }
    }
}

```

```

// Reverse the linked list
void reverseLinkedList(struct Node** head_ref) {
    struct Node* prev = NULL;
    struct Node* current = *head_ref;
    struct Node* next;

    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
}

// Concatenation of two linked lists
void concatenation(struct Node** head1_ref, struct Node** head2_ref) {
    struct Node* temp1 = *head1_ref;
    struct Node* temp2 = *head2_ref;

    // Move temp1 to the end of first list
    while (temp1->next != NULL)
        temp1 = temp1->next;

    // Connect the two lists
    temp1->next = temp2;
}

int main() {
    // Create the first linked list
    struct Node* head1 = NULL;
    insertAtBeginning(&head1, 10);
    insertAtBeginning(&head1, 20);
    insertAtBeginning(&head1, 30);
    printf("First Linked List: ");
    display(head1);
    // Create the second linked list
    struct Node* head2 = NULL;
    insertAtEnd(&head2, 40);
    insertAtEnd(&head2, 50);
    insertAtEnd(&head2, 60);
    printf("Second Linked List: ");
    display(head2);

    // Concatenate the two linked lists
    concatenation(&head1, &head2);
    printf("Concatenated Linked List: ");
    display(head1);
}

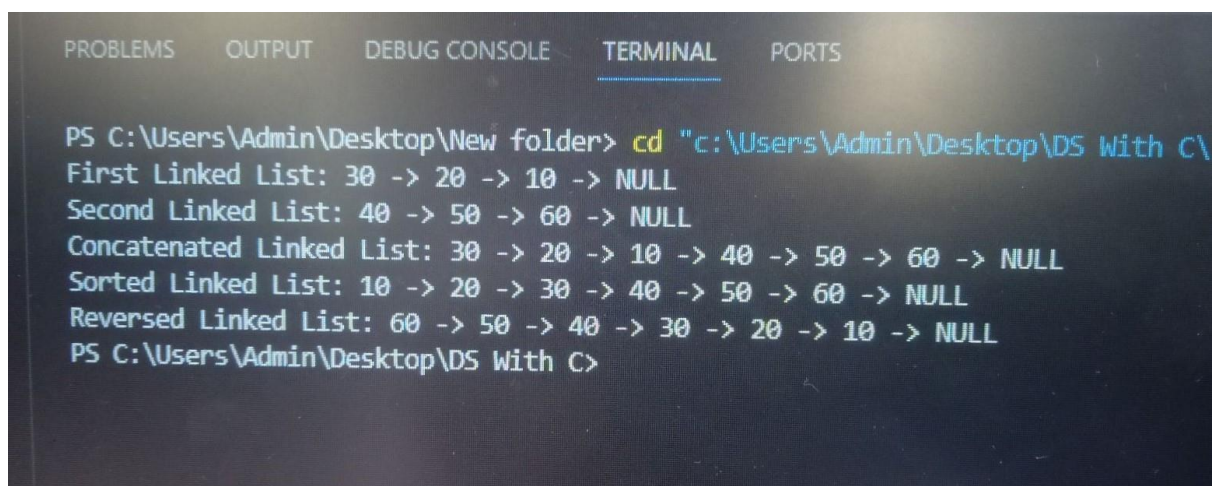
```

```
// Sort the concatenated linked list
sortLinkedList(&head1);
printf("Sorted Linked List: ");
display(head1);

// Reverse the sorted linked list
reverseLinkedList(&head1);
printf("Reversed Linked List: ");
display(head1);

return 0;
}
```

## OUTPUT



The screenshot shows a terminal window with the following output:

```
PS C:\Users\Admin\Desktop\New folder> cd "c:\Users\Admin\Desktop\DS With C\
First Linked List: 30 -> 20 -> 10 -> NULL
Second Linked List: 40 -> 50 -> 60 -> NULL
Concatenated Linked List: 30 -> 20 -> 10 -> 40 -> 50 -> 60 -> NULL
Sorted Linked List: 10 -> 20 -> 30 -> 40 -> 50 -> 60 -> NULL
Reversed Linked List: 60 -> 50 -> 40 -> 30 -> 20 -> 10 -> NULL
PS C:\Users\Admin\Desktop\DS With C>
```