LAB PROGRAM 6a

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;
};

struct node *head1 = NULL, *head2 = NULL;

struct node* create(struct node *head) {
    int n, x;
    struct node *temp, *newnode;

    printf("Enter number of nodes: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        newnode = (struct node*)malloc(sizeof(struct node));
        scanf("%d", &x);
        newnode->data = x;
        newnode->next = NULL;

        if (head == NULL) {
```

```
            head = newnode;
            temp = head;
        } else {
            temp->next = newnode;
            temp = newnode;
        }
    }
    return head;
}

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

struct node* sort(struct node *head) {
    struct node *i, *j;
    int t;

    for (i = head; i != NULL; i = i->next) {
        for (j = i->next; j != NULL; j = j->next) {
            if (i->data > j->data) {
                t = i->data;
                i->data = j->data;
                j->data = t;
```

```
            }
        }
    }
    return head;
}

struct node* reverse(struct node *head) {
    struct node *prev = NULL, *cur = head, *next;

    while (cur != NULL) {
        next = cur->next;
        cur->next = prev;
        prev = cur;
        cur = next;
    }
    return prev;
}

struct node* concatenate(struct node *h1, struct node *h2) {
    struct node *temp = h1;

    if (h1 == NULL)
        return h2;

    while (temp->next != NULL)
        temp = temp->next;

    temp->next = h2;
    return h1;
```

```
}

int main() {
    int choice;

    while (1) {
        printf("\n1.Create List 1\n2.Create List
2\n3.Sort\n4.Reverse\n5.Concatenate\n6.Display\n7.Exit\n");
        scanf("%d", &choice);

        switch (choice) {
            case 1: head1 = create(head1); break;
            case 2: head2 = create(head2); break;
            case 3: head1 = sort(head1); break;
            case 4: head1 = reverse(head1); break;
            case 5: head1 = concatenate(head1, head2); break;
            case 6: display(head1); break;
            case 7: exit(0);
            default: printf("Invalid choice\n");
        }
    }
}
```

OUTPUT:

```
1.Create List 1
2.Create List 2
3.Sort
4.Reverse
5.Concatenate
6.Display
7.Exit
1

Enter number of nodes:4

12

23

45

34
```

```
1.Create List 1
2.Create List 2
3.Sort
4.Reverse
5.Concatenate
6.Display
7.Exit
2

Enter number of nodes:3

12

11

15
```

```
1.Create List 1
2.Create List 2
3.Sort
4.Reverse
5.Concatenate
6.Display
7.Exit
3


1.Create List 1
2.Create List 2
3.Sort
4.Reverse
5.Concatenate
6.Display
7.Exit
4
```

```
1.Create List 1
2.Create List 2
3.Sort
4.Reverse
5.Concatenate
6.Display
7.Exit
5


1.Create List 1
2.Create List 2
3.Sort
4.Reverse
5.Concatenate
6.Display
7.Exit
6

45 -> 34 -> 23 -> 12 -> 12 -> 11 -> 15 -> NULL
```