Singly Linked List Sort, Reverse, and Concatenation

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

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

struct Node* createNode(int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = NULL;
   return newNode;
}

void insertEnd(struct Node** head, int value) {
   struct Node* newNode = createNode(value);

   if (*head == NULL) {
      *head = newNode;
      return;
   }

   struct Node* temp = *head;
   while (temp->next != NULL)
      temp = temp->next;

   temp->next = newNode;
```

```
}

void display(struct Node* head) {
   if (head == NULL) {
      printf("List is empty.\n");
      return;
   }

   printf("Linked List: ");
   while (head != NULL) {
      printf("%d -> ", head->data);
      head = head->next;
   }
   printf("NULL\n");
}

// Sort linked list (Bubble Sort)
void sortList(struct Node* head) {
   struct Node *i, *j;
   int temp;

   if (head == NULL)
      return;

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

```
        }
      }
    }
    printf("List sorted successfully.\n");
}

// Reverse linked list
void reverseList(struct Node** head) {
    struct Node *prev = NULL, *curr = *head, *next = NULL;

    while (curr != NULL) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    *head = prev;

    printf("List reversed successfully.\n");
}

// Concatenate two lists
struct Node* concatenate(struct Node* head1, struct Node* head2) {
    if (head1 == NULL) return head2;
    if (head2 == NULL) return head1;

    struct Node* temp = head1;
    while (temp->next != NULL)
        temp = temp->next;
```

```
    temp->next = head2;
    return head1;
}

int main() {
    struct Node *list1 = NULL, *list2 = NULL;
    int choice, value;

    printf("\n--- Linked List Operations ---\n");
    printf("1. Insert into List 1\n");
    printf("2. Insert into List 2\n");
    printf("3. Sort List 1\n");
    printf("4. Reverse List 1\n");
    printf("5. Concatenate List1 + List2\n");
    printf("6. Display List 1\n");
    printf("7. Display List 2\n");
    printf("8. Exit\n");

    while (1) {

        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value: ");
                scanf("%d", &value);
                insertEnd(&list1, value);
                break;
```

```
case 2:
   printf("Enter value: ");
   scanf("%d", &value);
   insertEnd(&list2, value);
   break;

case 3:
   sortList(list1);
   break;

case 4:
   reverseList(&list1);
   break;

case 5:
   list1 = concatenate(list1, list2);
   printf("Lists concatenated successfully.\n");
   break;

case 6:
   display(list1);
   break;

case 7:
   display(list2);
   break;

case 8:
   exit(0);

default:
```

```
            printf("Invalid choice!\n");
        }
    }

    return 0;
}
```

```
--- Linked List Operations ---
1. Insert into List 1
2. Insert into List 2
3. Sort List 1
4. Reverse List 1
5. Concatenate List1 + List2
6. Display List 1
7. Display List 2
8. Exit
Enter choice: 1
Enter value: 10
Enter choice: 1
```

```
Enter value: 20
Enter choice: 1
Enter value: 30
Enter choice: 1
Enter value: 40
Enter choice: 2
Enter value: 11
Enter choice: 2
Enter value: 21
Enter choice: 2
Enter value: 31
Enter choice: 2
```

```
Enter value: 41
Enter choice: 3
List sorted successfully.
Enter choice: 6
Linked List: 10 -> 20 -> 30 -> 40 -> NULL
Enter choice: 4
List reversed successfully.
Enter choice: 6
Linked List: 40 -> 30 -> 20 -> 10 -> NULL
Enter choice: 5
Lists concatenated successfully.
Enter choice: 6
```

```
List reversed successfully.
Enter choice: 6
Linked List: 40 -> 30 -> 20 -> 10 -> NULL
Enter choice: 5
Lists concatenated successfully.
Enter choice: 6
Linked List: 40 -> 30 -> 20 -> 10 -> 11 -> 21 -> 31 -> 41 -> NULL
Enter choice: 7
Linked List: 11 -> 21 -> 31 -> 41 -> NULL
Enter choice: 8
```