```
#include <stdio.h>
#include <stdlib.h>
// Structure for node
struct Node {
    int data;
    struct Node* next;
};
// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
// Insert node at end
void insertEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
}
// Display list
void display(struct Node* head) {
    while (head != NULL) {
```

```
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}
// Sort the linked list (Bubble Sort)
void sortList(struct Node* head) {
    struct Node *i, *j;
    int temp;
    for (i = head; i != 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;
            }
        }
    }
}
// Reverse the 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;
```

```
}
// Concatenate two linked lists
void concatenate(struct Node** head1, struct Node* head2) {
    if (*head1 == NULL) {
        *head1 = head2;
        return;
    }
    struct Node* temp = *head1;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = head2;
}
// Main function
int main() {
    struct Node* list1 = NULL;
    struct Node* list2 = NULL;
    // Creating first list
    insertEnd(&list1, 3);
    insertEnd(&list1, 1);
    insertEnd(&list1, 4);
    // Creating second list
    insertEnd(&list2, 6);
    insertEnd(&list2, 2);
    insertEnd(&list2, 5);
    printf("List 1: ");
    display(list1);
    printf("List 2: ");
    display(list2);
    // Sorting list1
```

```
    sortList(list1);
    printf("Sorted List 1: ");
    display(list1);
   // Reversing list1
    reverseList(&list1);
    printf("Reversed List 1: ");
    display(list1);
    // Concatenation
    concatenate(&list1, list2);
    printf("Concatenated List: ");
    display(list1);
    return 0;
}
```