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

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

typedef struct Node Node;

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

void append(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* current = *head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newNode;
    }
}

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

void sortList(Node** head) {
    if (*head == NULL) {
        return;
    }
```

```
    int temp;
    Node* current1 = *head;
    Node* current2;

    while (current1 != NULL) {
        current2 = current1->next;

        while (current2 != NULL) {
            if (current1->data > current2->data) {
                temp = current1->data;
                current1->data = current2->data;
                current2->data = temp;
            }

            current2 = current2->next;
        }

        current1 = current1->next;
    }
}

void reverseList(Node** head) {
    Node* prev = NULL;
    Node* current = *head;
    Node* nextNode;

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

    *head = prev;
}

void concatenateLists(Node** list1, Node* list2) {
    if (*list1 == NULL) {
        *list1 = list2;
    } else {
        Node* current = *list1;
        while (current->next != NULL) {
            current = current->next;
        }
```

```
        current->next = list2;
    }
}

int main() {
    Node* list1 = NULL;
    Node* list2 = NULL;

    append(&list1, 3);
    append(&list1, 1);
    append(&list1, 4);

    append(&list2, 2);
    append(&list2, 5);

    printf("Original List 1:\n");
    display(list1);

    printf("\nSorting List 1:\n");
    sortList(&list1);
    display(list1);

    printf("\nReversing List 1:\n");
    reverseList(&list1);
    display(list1);

    printf("\nOriginal List 2:\n");
    display(list2);

    printf("\nConcatenating List 1 and List 2:\n");
    concatenateLists(&list1, list2);
    display(list1);

    return 0;
}
```

Output:

```
Original List 1:
3 -> 1 -> 4 -> NULL

Sorting List 1:
1 -> 3 -> 4 -> NULL

Reversing List 1:
4 -> 3 -> 1 -> NULL

Original List 2:
2 -> 5 -> NULL

Concatenating List 1 and List 2:
4 -> 3 -> 1 -> 2 -> 5 -> NULL

Process returned 0 (0x0)   execution time : 0.059 s
Press any key to continue.
```