```
#include <stdio.h>
#include <stdlib.h>
//Represent a node of the singly linked list
struct node{
    int data;
    struct node *next;
};

//Represent the head and tail of the singly linked list
struct node *head, *tail = NULL;

//addNode() will add a new node to the list
void addNode(int data) {
    //Create a new node
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = data;
    newNode->next = NULL;

    //Checks if the list is empty
    if(head == NULL) {
        //If list is empty, both head and tail will point to new node
        head = newNode;
        tail = newNode;
    }
    else {
        //newNode will be added after tail such that tail's next will point to newNode
        tail->next = newNode;
        //newNode will become new tail of the list
        tail = newNode;
```

```
    }
}

    //sortList() will sort nodes of the list in ascending order
    void sortList() {
        //Node current will point to head
        struct node *current = head, *index = NULL;
        int temp;

        if(head == NULL) {
            return;
        }
        else {
            while(current != NULL) {
                //Node index will point to node next to current
                index = current->next;

                while(index != NULL) {
                    //If current node's data is greater than index's node data, swap the data
between them
                    if(current->data > index->data) {
                        temp = current->data;
                        current->data = index->data;
                        index->data = temp;
                    }
                    index = index->next;
                }
                current = current->next;
            }
```

```
        }
    }

//display() will display all the nodes present in the list
void display() {
    //Node current will point to head
    struct node *current = head;
    if(head == NULL) {
        printf("List is empty \n");
        return;
    }
    while(current != NULL) {
        //Prints each node by incrementing pointer
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

int main()
{
    //Adds data to the list
    addNode(9);
    addNode(7);
    addNode(2);
    addNode(5);
    addNode(4);

    //Displaying original list
```

```
    printf("Original list: \n");
    display();

    //Sorting list
    sortList();

    //Displaying sorted list
    printf("Sorted list: \n");
    display();

    return 0;
}
```