```c
/*Create a Singly Linked List of cars, insert at end, display, sort cars based
on decreasing order of mileage.
Cars has make, year, price, mileage.*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Define the structure for a car
typedef struct
{
    char make[50];
    int year;
    float price;
    float mileage;
} Car;
// Define the structure for a linked list node
typedef struct Node
{
    Car data;
    struct Node *next;
} Node;
// Define the structure for the linked list
typedef struct
{
    Node *start;
} LinkedList;
// Function to insert a car at the end of the linked list
void insertAtEnd(LinkedList *l, char make[], int year, float price, float
mileage)
{
    Node *newNode = (Node *)malloc(sizeof(Node));
    strcpy(newNode->data.make, make); // coz its a string
    newNode->data.year = year;        // after arrow its dot
    newNode->data.price = price;
    newNode->data.mileage = mileage;
    newNode->next = NULL;
    if (l->start == NULL)
    { // if LL empty
        l->start = newNode;
        return;
    }
    Node *current = l->start; // if some elements there,traverse to find last
element
    while (current->next != NULL)
    {
        current = current->next;
    }
    current->next = newNode;
}
```

```c
// Function to display the linked list of cars
void displayList(LinkedList l)
{
    Node *current = l.start;
    while (current != NULL)
    {
        printf("Make: %s\n", current->data.make);
        printf("Year: %d\n", current->data.year);
        printf("Price: %.2f\n", current->data.price);
        printf("Mileage: %.2f\n", current->data.mileage);
        printf("\n");
        current = current->next;
    }
}
void sort(LinkedList *l)
{
    int swapped;
    Node *traverse;
    if (l->start == NULL)
    {
        return; // Nothing to sort if the list is empty
    }
    do
    {
        swapped = 0;
        traverse = l->start;
        while (traverse->next != NULL)
        {                                                                //
doesnt reach last node till then traverse
            if (traverse->data.mileage < traverse->next->data.mileage)
//(i+1th)node ka mileage greater than ith node
            {
                // Swap the nodes' data (not the nodes themselves)
                Car temp = traverse->data; // initialize a temp variable of
type Car to store the data
                traverse->data = traverse->next->data;
                traverse->next->data = temp;
                swapped = 1; // to indicate
            }
            traverse = traverse->next; // update traverse pointer
        }
    } while (swapped);
}
int main()
{
    LinkedList l1;
    l1.start = NULL;
    // Insert cars at the end
```

```
    insertAtEnd(&l1, "Ferrari", 2020, 500000.0, 20.5);
    insertAtEnd(&l1, "Audi", 2019, 40000.0, 25);
    insertAtEnd(&l1, "Nissan", 2021, 50000.0, 29.4);
    printf("List of Cars:\n");
    displayList(l1);
    printf("Sorted list:\n");
    sort(&l1);
    displayList(l1);
    return 0;
}
```

```
List of Cars:
Make: Ferrari
Year: 2020
Price: 500000.00
Mileage: 20.50

Make: Audi
Year: 2019
Price: 40000.00
Mileage: 25.00

Make: Nissan
Year: 2021
Price: 50000.00
Mileage: 29.40

Sorted list:
Make: Nissan
Year: 2021
Price: 50000.00
Mileage: 29.40

Make: Audi
Year: 2019
Price: 40000.00
Mileage: 25.00

Make: Ferrari
Year: 2020
Price: 500000.00
Mileage: 20.50

PS C:\Users\Mark Lopes>
```