Code:

Part 1:

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

/**
* References: https://secweb.cs.odu.edu/~zeil/cs361/web
* /website/Lectures/sortingAndSearching/pages/orderedInsert.html
*/

typedef struct Car {
    char car_make[50];
    char car_model[50];
    int year;
    char color[50];
} Car;

// Predeclare functions
int ordered_insert(Car ** car_array, int last, Car * car);
int insert_sorted_array(Car ** car_array, int * arr_length);
void print_cars_array(Car ** car_array, int * arr_length);
void sort_cars_by_year(Car ** car_array, int * arr_length);
void print_duplicates(Car ** car_array, int * arr_length);
int PrintMenu();
int Run();
```

```
/**
 * Insert car into an array such that arr[first..last] is
 * sorted, given that arr[first..last-1] is already sorted.
 *
 * @param Car **, an array of car pointers.
 * @param Number, the amount of car pointers in the array of cars.
 * @param Car, the car being inserted.
 * @return Number, return the index where the car was sorted into.
 */
int ordered_insert(Car ** car_array, int last, Car * car)
{
    /**
    * Loop until a correct location is found for the car in the array
    */
    int i = last;
    while ((i > 0) && (strncmp(car->car_make, car_array[i-1]->car_make, 50) < 0))
    {
        car_array[i] = car_array[i-1];
        i = i - 1;
    }
    car_array[i] = car; // Insert the car into the appropriate location
    return i; // Return location placed at
}
```

CarRecords.txt hw1_part2.c hw1_part1.c

```c
/**
 * Provide a function insert_sorted_array( ) that reads and
 * stores the 10 records from file in an a sorted array of
 * structs. The array should be sorted based on the car make.
 * Each new record from file should be inserted into its correct
 * sorted location by shifting other records if necessary.
 *
 * @param Car **, an array of car pointers.
 * @param Number, the amount of car pointers in the array of cars.
 * @return Return 0 on no error
 */
int insert_sorted_array(Car ** car_array, int * arr_length)
{
    /**
     * Open the file to read car records from.
     */
    FILE * file;
    file = fopen("./CarRecords.txt", "r");
    if (file == NULL) {
        perror("Error opening the file './CarRecords.txt'");
        return -1;
    }

    // Add car records from the file until no more can be added
    // or the limit of 10 is reached.
    while(true) {
        Car * car = (Car *) malloc(sizeof(Car));
        if (fscanf(file, "%49[^,], %49[^,], %d, %49[^,\r\n]\r\n", car->car_make,
                car->car_model, &car->year, car->color) == 4 && *arr_length < 10) {
            ordered_insert(car_array, *arr_length, car);
            (*arr_length)++; // Increase the count of cars in the array
        } else {
            free(car); // Free the car not used and exit the loop
            break;
        }
    }

    return 0;
}
```

```c
/**
 * Print the cars in the array of Cars.
 *
 * @param Car **, an array of car pointers.
 * @param Number, the amount of car pointers in the array of cars.
 */
void print_cars_array(Car ** car_array, int * arr_length)
{
    /**
     * Loop through all the car records in the array and print them
     */
    int i;
    for(i = 0; i < *arr_length; i++) {
        printf("%s %s %d %s \n", car_array[i]->car_make, car_array[i]->car_model,
```

```c
 * Sort the cars by year in descending order
 *
 * @param Car **, an array of car pointers.
 * @param Number, the amount of car pointers in the array of cars.
 */
void sort_cars_by_year(Car ** car_array, int * arr_length) {

    /**
     * Bubble sort the cars by year.
     */
    Car * swap;
    int i,j;
    for (i = 0 ; i < ( *arr_length - 1 ); i++) {
        for (j = 0 ; j < *arr_length - i - 1; j++) {
            if (car_array[j]->year < car_array[j+1]->year) {
                swap           = car_array[j];
                car_array[j]   = car_array[j+1];
                car_array[j+1] = swap;
            }
        }
    }
}
```

```
/**
 * Print the duplicates in the array of cars
 *
 * @param Car **, an array of car pointers.
 * @param Number, the amount of car pointers in the array of cars.
 */
void print_duplicates(Car ** car_array, int * arr_length)
{
    /**
    * Brutce force method.. For each car check if there is another car
    * in the array of cars that matches it.
    */
    int i, j;
    for(i = 0; i < *arr_length; i++) {
        for (j = i + 1; j < *arr_length; j++) {
            if (strncmp(car_array[i]->car_make, car_array[j]->car_make, 50) == 0 &&
                strncmp(car_array[i]->car_model, car_array[j]->car_model, 50) == 0 &&
                car_array[i]->year == car_array[j]->year &&
                strncmp(car_array[i]->color, car_array[j]->color, 50) == 0) {

                // Print the car out if it exists elsewhere.
                printf("%s %s %d %s \n", car_array[i]->car_make, car_array[i]->car_model,
                                    car_array[i]->year, car_array[i]->color);
            }
        }
    }
}

```

```
/**
 * Print the menu and get a selection from the user.
 *
 * @return Number of selection.
 */
int PrintMenu()
{
    int sel;

    printf("Main menu:\n\n" );
    printf("1. Print the cars array\n" );
    printf("2. Insert the car records into a sorted array\n" );
    printf("3. Sort cars by year\n" );
    printf("4. Print duplicates\n" );
    printf("5. Exit \n" );
    printf("Select an option: " );

    scanf("%d", &sel); // Scan a digit from the user

    return sel; // Return the chosen digit
}
```

```
/**
 * Run the navigational loop
 *
 * @return Number on exit. 0 for no errors.
 */
int Run()
{
    int sel;
    int arr_length = 0;
    Car ** car_array = (Car **) malloc(10 * sizeof(struct Car *));

    // While true
    while(true) {

        // Print the menu and get a selection
        sel = PrintMenu();

        // Next step depends on the selection made
        switch(sel) {

            // User chose 1
            case 1:
                printf("You selected \"Print the cars array\"\n\n");
                print_cars_array(car_array, &arr_length);
                break;

            // User chose 2
            case 2:
                printf("You selected \"Insert the car records into a sorted array\"\n\n");
                insert_sorted_array(car_array, &arr_length);
                break;
```

```
            // User chose 3
            case 3:
                printf("You selected \"Sort cars by year\"\n\n");
                sort_cars_by_year(car_array, &arr_length);

                break;

            // User chose 4
            case 4:
                printf("You selected \"Print duplicates\"\n\n");
                print_duplicates(car_array, &arr_length);
                break;

            // User chose 5
            case 5:
                printf("You selected \"Exit\"\n\n");

                // Return here, with no erros, to exit the function.
                // Clean up will be next
                return 0;

            // User chose soomething not on the menu
            default:
                printf("Please enter a valid number from the menu!\n\n");
                break;
        }

    printf("--------------------\n");

    }
}
```

```
/**
 * Provides two functions that can compute the sum
 * or the difference using bitwise logical operators.
 *
 * @return Number on exit. 0 for no errors.
 */
int main()
{
    Run();
    return 0;
} //end main
```

Part 2:

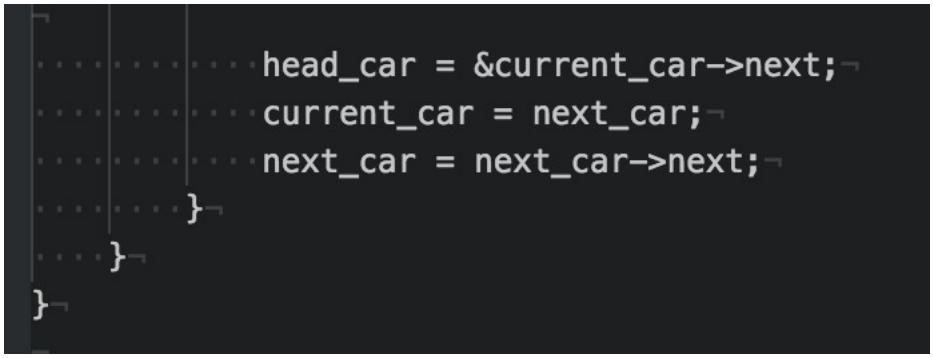
```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>

/**
* References:
* http://www.geeksforgeeks.org/given-a-linked-
* list-which-is-sorted-how-will-you-insert-in-sorted-way/
*
* http://stackoverflow.com/questions/21388916/
* bubble-sort-singly-linked-list-in-c-with-pointers
*/

typedef struct Car {
    char car_make[50];
    char car_model[50];
    int year;
    char color[50];
    struct Car * next;
} Car;

// Predeclare functions
void print_cars_array(Car * head);
void ordered_insert(Car ** head, Car * new_car);
int insert_sorted_array(Car ** head, int * count);
void sort_cars_by_year(Car ** head);
void print_duplicates(Car * head);
int PrintMenu();
int Run();
```

```c
/**
 * Insert car into a linked list such that list[first..last] is
 * sorted, given that list[first..last-1] is already sorted
 *
 * @param Car ** head, a reference to the head of the linked list
 */
void ordered_insert(Car ** head, Car * new_car)
{
    Car * current;
    // Special case for the head
    if (*head == NULL || (strncmp((*head)->car_make, new_car->car_make, 50) >= 0))
    {
        new_car->next = *head;
        *head = new_car;
    }
    else
    {
        // Locate the node before the point of insertion
        current = *head;
        while (current->next != NULL &&
               (strncmp(current->next->car_make, new_car->car_make, 50) < 0))
               {
                   current = current->next;
               }
               new_car->next = current->next;
               current->next = new_car;
        }
}
```

```c
/**
 * Reads the 10 records from the file into a sorted singly linked list.
 * The linked list should be sorted based on the car model. Each ne
 * record from file should be inserted into its correct sorted location
 * in the linked list.
 *
 * @param Car ** head, a reference to the head of the linked list
 * @return Number, return 0 for no error
 */
int insert_sorted_array(Car ** head, int * count)
{
    /**
    * Open the file to read car records from.
    */
    FILE * file;
    file = fopen("./CarRecords.txt", "r");
    if (file == NULL) {
        perror("Error opening the file './CarRecords.txt'");
        return -1;
    }

    // Add car records from the file to the list until no more can be added
    // or the limit of 10 is reached.
    while(true) {
        Car * car = (Car *) malloc(sizeof(Car));
        car -> next = NULL;
        if (fscanf(file, "%49[^,], %49[^,], %d, %49[^,\r\n]\r\n", car->car_make,
                car->car_model, &car->year, car->color) == 4 && *count < 10) {
                    ordered_insert(head, car);
                    (*count)++;
                } else {
                    free(car); // Free the car not used and exit the loop
                    break;
                }
    }

    return 0;
}
```

```
/**
 * Print the cars in the list
 *
 * @param Car *, a pointer to the head of the list
 */
void print_cars_array(Car * head)
{
    Car * temp = head;
    while(temp != NULL)
    {
        printf("%s %s %d %s \n", temp->car_make, temp->car_model,
                temp->year, temp->color); // Print the data
        temp = temp->next;
    }
}
```

```
/**
 * Sort the linked list of cars by year using bubble sort. The sort
 * relies on the pointers being sorted.
 *
 * @param Car ** head, a reference to the head of the linked list
 */
void sort_cars_by_year(Car ** head)
{
    int done = 0; // True if no swaps were made in a pass

    /**
    * Return if head is NULL or the next element is NULL
    */
    if (*head == NULL || (*head)->next == NULL)
        return;

    /**
    * While the sorting is not finished
    */
    while (!done) {
        Car **head_car = head; // Set head reference
        Car *current_car = *head; // Current car, iterator pointer
        Car *next_car = (*head)->next; // Next car, next pointer

        done = 1;

        /**
        * Loop through the linked list
        */
        while (next_car != NULL) {
            if (strcmp(current_car->color, next_car->color) > 0) {
                current_car->next = next_car->next;
                next_car->next = current_car;
                *head_car = next_car;

                done = 0;
            }
```



```
            head_car = &current_car->next;
            current_car = next_car;
            next_car = next_car->next;
        }
    }
}
```

```c
/**
 * Print the duplicates in the linked list of cars
 *
 * @param Car * head, the head of the linked list
 */
void print_duplicates(Car * head)
{
    Car * first = head;

    if (first == NULL) // Return if head is NULL
        return;

    while (first != NULL)
    {
        Car * second = first->next;

        /**
        * Brutce force method.. For each car check if there is another car
        * in the linked list of cars that matches it.
        */
        while (second != NULL)
        {
            /**
            * Compare first node with second node. Compare all the data to see
            * if equality is true.
            */
            if (strncmp(first->car_make, second->car_make, 50) == 0 &&
            strncmp(first->car_model, second->car_model, 50) == 0 &&
            first->year == second->year &&
            strncmp(first->color, second->color, 50) == 0)
            {
                printf("%s %s %d %s \n", first->car_make, first->car_model,
                        first->year, first->color); // Print duplicate
            }

            second = second->next; // Increment the second node
        }

        first = first->next; // Increment the first node
    }
```

```c
/**
 * Print the menu and get a selection from the user.
 *
 * @return Number of selection.
 */
int PrintMenu()
{
    int sel;

    printf("Main menu:\n\n" );
    printf("1. Print the cars list\n" );
    printf("2. Insert the car records into a sorted linked list\n" );
    printf("3. Sort cars by color\n" );
    printf("4. Print duplicates\n" );
    printf("5. Exit \n" );
    printf("Select an option: " );

    scanf("%d", &sel); // Scan a digit from the user

    return sel; // Return the chosen digit
}
```

```
            // User chose 4
            case 4:
                printf("You selected \"Print duplicates\"\n\n");
                print_duplicates(head);
                break;

            // User chose 5
            case 5:
                printf("You selected \"Exit\"\n\n");

                // Return here, with no erros, to exit the function.
                // Clean up will be next
                return 0;

            // User chose soomething not on the menu
            default:
                printf("Please enter a valid number from the menu!\n\n");
                break;
        }

    printf("--------------------\n");

    }
}
```

```
/**
 * Provides two functions that can compute the sum
 * or the difference using bitwise logical operators.
 *
 * @return Number on exit. 0 for no errors.
 */
int main()
{
    Run();
    return 0;
} //end main
```

Running the Code:

Part 1:

```
Main menu:

1. Print the cars array
2. Insert the car records into a sorted array
3. Sort cars by year
4. Print duplicates
5. Exit
Select an option: 2
You selected "Insert the car records into a sorted array"

--------------------
Main menu:

1. Print the cars array
2. Insert the car records into a sorted array
3. Sort cars by year
4. Print duplicates
5. Exit
Select an option: 1
You selected "Print the cars array"

Dodge Neon 1993 pink
Ford Fusion 2013 yellow
Ford Expedition 2009 silver
Ford Fusion 2013 yellow
Honda Fit 2015 blue
Jeep Cherokee 1999 red
Mazda Protoge 1996 gold
Subaru Outback 2016 green
Toyota Corolla 2006 white
Toyota Corolla 2006 white
--------------------
Main menu:

1. Print the cars array
2. Insert the car records into a sorted array
3. Sort cars by year
4. Print duplicates
5. Exit
Select an option:
```

```
Select an option: 4
You selected "Print duplicates"

Ford Fusion 2013 yellow
Toyota Corolla 2006 white
--------------------
```

```
5. Exit
Select an option: 3
You selected "Sort cars by year"

--------------------
```

```

Subaru Outback 2016 green
Honda Fit 2015 blue
Ford Fusion 2013 yellow
Ford Fusion 2013 yellow
Ford Expedition 2009 silver
Toyota Corolla 2006 white
Toyota Corolla 2006 white
Jeep Cherokee 1999 red
Mazda Protoge 1996 gold
Dodge Neon 1993 pink
--------------------
```

```
Select an option: 5
You selected "Exit"
```

Part 2:

```
Main menu:

1. Print the cars list
2. Insert the car records into a sorted linked list
3. Sort cars by color
4. Print duplicates
5. Exit
Select an option: 2
You selected "Insert the car records into a sorted linked list"

--------------------
Main menu:

1. Print the cars list
2. Insert the car records into a sorted linked list
3. Sort cars by color
4. Print duplicates
5. Exit
Select an option: 1
You selected "Print the cars list"

Dodge Neon 1993 pink
Ford Fusion 2013 yellow
Ford Expedition 2009 silver
Ford Fusion 2013 yellow
Honda Fit 2015 blue
Jeep Cherokee 1999 red
Mazda Protoge 1996 gold
Subaru Outback 2016 green
Toyota Corolla 2006 white
Toyota Corolla 2006 white
--------------------
```

```
5. Exit
Select an option: 4
You selected "Print duplicates"

Ford Fusion 2013 yellow
Toyota Corolla 2006 white
--------------------
```

```
--------------------
Main menu:

1. Print the cars list
2. Insert the car records into a sorted linked list
3. Sort cars by color
4. Print duplicates
5. Exit
Select an option: 3
You selected "Sort cars by color"

--------------------
Main menu:

1. Print the cars list
2. Insert the car records into a sorted linked list
3. Sort cars by color
4. Print duplicates
5. Exit
Select an option: 1
You selected "Print the cars list"

Honda Fit 2015 blue
Mazda Protoge 1996 gold
Subaru Outback 2016 green
Dodge Neon 1993 pink
Jeep Cherokee 1999 red
Ford Expedition 2009 silver
Toyota Corolla 2006 white
Toyota Corolla 2006 white
Ford Fusion 2013 yellow
Ford Fusion 2013 yellow
--------------------
```