DUY VU

EECE2160

HOMEWORK 1

a)

HW1.c

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>


//struct carrecord
struct CarRecord
{
    char make[20];
    char model[20];
    int year;
    char color[20];
};


//insert the records into an array passed from main
void insert_array( struct CarRecord carRecords[10] ) {
    printf("\nInserting Records from CarRecords.txt...\n");
    //file pointer
    FILE *fp;
    char temp[20];
    int i = 0;
    //open file
    fp = fopen("CarRecords.txt","r"); // read mode
    //check if error while openning file
    if ( fp == NULL )
    {
        perror("Error while opening the file.\n");
        exit(EXIT_FAILURE);
    }


    //scan the record line by line and insert them to the appropriate variables
    while (fscanf(fp, "%s %s %s %s", carRecords[i].make,
                  carRecords[i].model, temp, carRecords[i].color) != EOF) {
        //convert year from string to number
        carRecords[i].year = atoi(strncpy(temp, temp, strlen(temp)-1));
        //delete the last comma in make and model
        carRecords[i].make[strlen(carRecords[i].make) - 1] = '\0';
        carRecords[i].model[strlen(carRecords[i].model) - 1] = '\0';
        i++;
    }
}


//print the array of car records
void print_cars_array( struct CarRecord carRecords[10] ) {
    int i;
    printf("\nPrinting car records:\n");
    //print all the records one by one
    for (i = 0; i < 10; i++) {
        printf("Record %d:\tMake: %s,\tModel: %10s,\tYear: %d,\tColor: %s\n",
               i + 1, carRecords[i].make, carRecords[i].model, carRecords[i].year,
carRecords[i].color);
    }
}


//sort the cars by year
void sort_cars_by_year( struct CarRecord carRecords[10] ) {
    int i, j;
    //bubble sort
    for (i = 0; i < 10; i++) {
        for (j = 1; j < 10 - i; j++) {
            //compare between the 2 years
            if (carRecords[j - 1].year > carRecords[j].year) {
                //make a temp record for swapping
                struct CarRecord temp = carRecords[j];
                //swap the first with the second
                carRecords[j] = carRecords[j - 1];
                //swap the second with temp
                carRecords[j - 1] = temp;
            }
        }
    }
}


//print the duplicated records
```

```c
void print_duplicates( struct CarRecord carRecords[10] ) {
    int i, j;
    //go through the list
    //compare the element with all the ones behind it
    for (i = 0; i < 10; i++) {
        for (j = i + 1; j < 10; j++) {
            //compare the 2 records
            if (strcmp(carRecords[i].make, carRecords[j].make) == 0
                    && strcmp(carRecords[i].model, carRecords[j].model) == 0
                    && carRecords[i].year == carRecords[j].year
                    && strcmp(carRecords[i].color, carRecords[j].color) == 0) {
                //then print them both
                printf("Record %d:\tMake: %s,\tModel: %10s,\tYear: %d,\tColor: %s\n", i
+ 1, carRecords[i].make,
                        carRecords[i].model, carRecords[i].year, carRecords[i].color);
                printf("Record %d:\tMake: %s,\tModel: %10s,\tYear: %d,\tColor: %s\n", j
+ 1, carRecords[j].make,
                        carRecords[j].model, carRecords[j].year, carRecords[j].color);
                printf("-----------------------------------\n");
            }
        }
    }
}



//main function
int main()
{
    //save user's options
    int option;
    //array of car's records
    struct CarRecord carRecords[10];
    //wait for user to choose
    while (option != 5) {
        printf("\n\nMENU - Select an option:\n\n");
        printf("1. Print the cars array\n");
        printf("2. Insert car records into a sorted array\n");
        printf("3. Sort cars by year\n");
        printf("4. Print duplicates\n");
        printf("5. Exit\n\n");
        printf("Select an option: ");
        //getting user's option input
        scanf("%d", &option);
        //switch cases
        switch (option) {
        case 1 :
            printf("You selected \"Print the car records\"\n");
            print_cars_array(carRecords);
            break;
        case 2 :
            printf("You selected \"Insert the records from file\"\n");
            insert_array(carRecords);
            break;
        case 3 :
            printf("You selected \"Sort the records by year\"\n");
            sort_cars_by_year(carRecords);
            break;
        case 4 :
            printf("You selected \"Print duplicated records\"\n");
            print_duplicates(carRecords);
            break;
        case 5 :
            printf("Exiting...\n\n\n");
            break;
        default:
            printf("Invalid Option");
            break;
        }
        printf("\nDone\n\n");
    }
    return 0;
}
```

```
-bash-4.3$ gcc HW1FIXED.c
-bash-4.3$ ./a.out


MENU - Select an option:

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

Select an option: 2
You selected "Insert the records from file"

Inserting Records from CarRecords.txt...

Done



MENU - Select an option:

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

Select an option: 1
You selected "Print the car records"

Printing car records:
Record 1:       Make: Subaru,   Model:    Outback,      Year: 2016,     Color: green
Record 2:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 3:       Make: Dodge,    Model:       Neon,      Year: 1993,     Color: pink
Record 4:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
Record 5:       Make: Honda,    Model:        Fit,      Year: 2015,     Color: blue
Record 6:       Make: Ford,     Model: Expedition,      Year: 2009,     Color: silver
Record 7:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 8:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
Record 9:       Make: Jeep,     Model:   Cherokee,      Year: 1999,     Color: red
Record 10:      Make: Mazda,    Model:    Protoge,      Year: 1996,     Color: gold

Done
```

```
MENU - Select an option:

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

Select an option: 3
You selected "Sort the records by year"

Done



MENU - Select an option:

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

Select an option: 1
You selected "Print the car records"

Printing car records:
Record 1:       Make: Dodge,    Model:       Neon,      Year: 1993,     Color: pink
Record 2:       Make: Mazda,    Model:    Protoge,      Year: 1996,     Color: gold
Record 3:       Make: Jeep,     Model:   Cherokee,      Year: 1999,     Color: red
Record 4:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 5:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 6:       Make: Ford,     Model: Expedition,      Year: 2009,     Color: silver
Record 7:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
Record 8:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
Record 9:       Make: Honda,    Model:        Fit,      Year: 2015,     Color: blue
Record 10:      Make: Subaru,   Model:    Outback,      Year: 2016,     Color: green

Done
```

```


MENU - Select an option:

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

Select an option: 4
You selected "Print duplicated records"
Record 4:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 5:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
-----------------------------------
Record 7:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
Record 8:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
-----------------------------------

Done
```

b)

HW1LINKEDLIST.c

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



//struct CarRecord
struct CarRecord
{
    char make[20];
    char model[20];
    int year;
    char color[20];
    struct CarRecord *next;
};



//struct List
struct List
{
    // First car in the list. A value equal to NULL indicates that the
    // list is empty.
    struct CarRecord *head;
    // Current car in the list. A value equal to NULL indicates a
    // past-the-end position.
    struct CarRecord *current;
    // Pointer to the element appearing before 'current'. It can be NULL if
    // 'current' is NULL, or if 'current' is the first element in the list.
    struct CarRecord *previous;
    // Number of persons in the list
    int count;
};



// Give an initial value to all the fields in the list.
void ListInitialize(struct List *list)
{
    list->head = NULL;
    list->current = NULL;
    list->previous = NULL;
    list->count = 0;
}



// Move the current position in the list one element forward. If last element
// is exceeded, the current position is set to a special past-the-end value.
void ListNext(struct List *list)
{
    if (list->current)
    {
        list->previous = list->current;
        list->current = list->current->next;
    }
}



// Move the current position to the first element in the list.
void ListHead(struct List *list)
{
    list->previous = NULL;
    list->current = list->head;
}



// Get the element at the current position, or NULL if the current position is
// past-the-end.
struct CarRecord *ListGet(struct List *list)
{
    return list->current;
}



// Insert a car before the element at the current position in the list. If
// the current position is past-the-end, the car is inserted at the end of
// the list. The new car is made the new current element in the list.
void ListInsert(struct List *list, struct CarRecord *car)
{
```

```
    // Set 'next' pointer of current element
    car->next = list->current;
    // Set 'next' pointer of previous element. Treat the special case where
    // the current element was the head of the list.
    if (list->current == list->head)
        list->head = car;
    else
        list->previous->next = car;
    // Set the current element to the new car
    list->current = car;
}
```

```
//Print the struct
void PrintCarRecord(struct CarRecord *carRecord)
{
    printf("\tMake: %s,\tModel: %10s,\tYear: %d,\tColor: %s\n",
           carRecord->make, carRecord->model,
           carRecord->year, carRecord->color);
}
```

```
// takes the list pointer and uses print person to print each person
void print_cars_list(struct List *list)
{
    int i = 0;
    //start the current form the head pointer
    ListHead(list);
    //move curent pointer through the list and print each Person link
    while (list->current)
    {
        printf("Record %d: ", i + 1);
        //print the car record at list->current
        PrintCarRecord(ListGet(list));
        //move list current to the next record
        ListNext(list);
        i++;
    }
    //return the current pointer to the beginning
    ListHead(list);
}
```

```
//swap the position of the 2 person node
//this function is used for sorting the linked list using buble sort
void Swap(struct List *list)
{
    //make temp pointer for swaping
    struct CarRecord *temp;


    //if the current is head, then we need to mark the next as head
    //before swap the current head
    if (list->current == list->head)
    {
        list->head = list->current->next;
    }


    //current node is 1st, next of it is 2nd, and next of its next is 3rd
    //0th is the node previous to current
    //temp point to the next node (2nd node)
    temp = list->current->next;
    //point the current node 1st to the (3rd node)
    list->current->next = list->current->next->next;
    //connect the 0th node to the 2nd node
    if (list->previous)
        list->previous->next = temp;


    //point the (2nd node) to the current node 1st
    temp->next=list->current;
    //point previous pointer to the prviously 2nd node
    list->previous = temp;
}
```

```
//bubble sort the linked list
void sort_cars_by_color(struct List *list)
{
    int i;
    //place current pointer at head
    ListHead(list);
    //bubble sort
    for (i=0;i<list->count-1;i++)
    {
        while (list->current->next)
        {
            //compare the 2 colors
            if (strcmp(list->current->color, list->current->next->color)>0)
            {
                //swap
                Swap(list);
            }
            else
            {
```

```
                //if can't swap then move to the next node
                //and continute comparing
                ListNext(list);
            }
        }
        //return to the head to compare again
        ListHead(list);
    }
}



//Add a car to the linked list
void insert_linkedList(struct List *list)
{
    //freeing the old memory everytime inserting is called
    //starting with the head, then move along the list
    ListHead(list);
    ListNext(list);
    //remove the previous until reach the end
    while (list->current) {
        free(list->previous);
        ListNext(list);
    }
    //free the last one
    free(list->current);


    //initialize the new list
    ListInitialize(list);


    printf("\nInserting Records from CarRecords.txt...\n");
    FILE *fp;
    char temp[20];
    int i = 0;
    //reading from file
    fp = fopen("CarRecords.txt","r"); // read mode
    //check for openning file error
    if ( fp == NULL )
    {
        perror("Error while opening the file.\n");
        exit(EXIT_FAILURE);
    }
    //inserting from file to struct one by one
    for (; i < 10; i++)
    {
        //make a new car's record
        struct CarRecord *car = (struct CarRecord *)malloc(sizeof(struct CarRecord));
        //inserting from file to varialbles
        fscanf(fp, "%s %s %s %s", car->make, car->model, temp, car->color);
        //convert year from string to integer
        car->year = atoi(strncpy(temp, temp, strlen(temp)-1));
        //remove the last ending comma in make
        car->make[strlen(car->make) - 1] = '\0';
        //remove the last ending comma in model
        car->model[strlen(car->model) - 1] = '\0';
        //set up the car pointers
        car->next = NULL;
        //insert the car
        ListInsert(list, car);
        //adjusting the count of the list
        list->count = list->count + 1;
    }
}



//print_duplicates
void print_duplicates(struct List *list)
{
    int i;
    struct CarRecord *cursor = list->head->next;
    //place current pointer at head
    ListHead(list);
    //compare the current car with the rest of the list behind
    //which cursor is being used to point to the comparing target
    for (i=0;i<list->count-1;i++)
    {
        while (cursor)
        {
            //compare the 2 cars
            if (strcmp(list->current->make, cursor->make)==0
                    && strcmp(list->current->model, cursor->model)==0
                    && list->current->year == cursor->year
                    && strcmp(list->current->color, cursor->color)==0)
            {
                //then print them both if the same
                PrintCarRecord(list->current);
                PrintCarRecord(cursor);
            }
            //check the next one
            cursor = cursor->next;
        }
        //move the current pointer
        ListNext(list);
        //move the cursor
        cursor = list->current->next;
    }
}



int main()
{
```

```c
int option;
struct List list;                       // Create the main list
ListInitialize(&list);                  // Initialize the list
while (option != 5)
{
    printf("\n\nMENU - Select an option:\n\n");
    printf("1. Print the cars list\n");
    printf("2. Insert car records into a sorted list\n");
    printf("3. Sort cars by color\n");
    printf("4. Print duplicates\n");
    printf("5. Exit\n\n");
    printf("Select an option: ");
    //getting user's option input
    scanf("%d", &option);
    //switch cases
    switch (option)
    {
    case 1 :
        printf("You selected \"Print the car records\"\n");
        print_cars_list(&list);
        break;
    case 2 :
        printf("You selected \"Insert the records from file\"\n");
        insert_linkedList(&list);
        break;
    case 3 :
        printf("You selected \"Sort the records by color\"\n");
        sort_cars_by_color(&list);
        break;
    case 4 :
        printf("You selected \"Print duplicated records\"\n");
        print_duplicates(&list);
        break;
    case 5 :
        printf("Exiting...\n\n\n");
        //Finalize();
        break;
    default:
        printf("Invalid Option");
        break;
    }
    printf("\nDone\n\n");
}
return 0;
}
```

```
MENU - Select an option:

1. Print the cars list
2. Insert car records into a sorted list
3. Sort cars by color
4. Print duplicates
5. Exit

Select an option: 3
You selected "Sort the records by color"

Done



MENU - Select an option:

1. Print the cars list
2. Insert car records into a sorted list
3. Sort cars by color
4. Print duplicates
5. Exit

Select an option: 1
You selected "Print the car records"
Record 1:       Make: Honda,    Model:        Fit,      Year: 2015,     Color: blue
Record 2:       Make: Mazda,    Model:    Protoge,      Year: 1996,     Color: gold
Record 3:       Make: Subaru,   Model:    Outback,      Year: 2016,     Color: green
Record 4:       Make: Dodge,    Model:       Neon,      Year: 1993,     Color: pink
Record 5:       Make: Jeep,     Model:   Cherokee,      Year: 1999,     Color: red
Record 6:       Make: Ford,     Model: Expedition,      Year: 2009,     Color: silver
Record 7:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 8:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 9:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
Record 10:      Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow

Done
```

```
MENU - Select an option:

1. Print the cars list
2. Insert car records into a sorted list
3. Sort cars by color
4. Print duplicates
5. Exit

Select an option: 4
You selected "Print duplicated records"
        Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
        Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
        Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
        Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow

Done



MENU - Select an option:

1. Print the cars list
2. Insert car records into a sorted list
3. Sort cars by color
4. Print duplicates
5. Exit

Select an option: 5
Exiting...
```

```
-bash-4.3$ gcc HW1LINKEDLIST.c
-bash-4.3$ ./a.out


MENU - Select an option:

1. Print the cars list
2. Insert car records into a sorted list
3. Sort cars by color
4. Print duplicates
5. Exit

Select an option: 2
You selected "Insert the records from file"

Inserting Records from CarRecords.txt...

Done



MENU - Select an option:

1. Print the cars list
2. Insert car records into a sorted list
3. Sort cars by color
4. Print duplicates
5. Exit

Select an option: 1
You selected "Print the car records"
Record 1:       Make: Mazda,    Model:    Protoge,      Year: 1996,     Color: gold
Record 2:       Make: Jeep,     Model:   Cherokee,      Year: 1999,     Color: red
Record 3:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
Record 4:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 5:       Make: Ford,     Model: Expedition,      Year: 2009,     Color: silver
Record 6:       Make: Honda,    Model:        Fit,      Year: 2015,     Color: blue
Record 7:       Make: Ford,     Model:     Fusion,      Year: 2013,     Color: yellow
Record 8:       Make: Dodge,    Model:       Neon,      Year: 1993,     Color: pink
Record 9:       Make: Toyota,   Model:    Corolla,      Year: 2006,     Color: white
Record 10:      Make: Subaru,   Model:    Outback,      Year: 2016,     Color: green

Done
```

c)

HW1Part3.cpp

```
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <iostream>
#include <stdlib.h>
#include <sys/mman.h>
#include <fstream>
#include <string>


using namespace std;


//class Car
class Car {
private:
    //car's properties
    string make;
    string model;
    int year;
    string color;
public:
    //constructor
    Car() {
        make = "";
        model = "";
        year = 0;
        color = "";
    }
    //set the fields to the desired values
    void setFields(string mk, string md, int yr,string cl) {
        make = mk;
        model = md;
        year = yr;
        color = cl;
    }
    //return make
    string getMake() {
        return make;
    }
    //return model
    string getModel() {
        return model;
    }
    //return year
    int getYear() {
        return year;
    }
    //get color
    string getColor() {
        return color;
    }
};



//class CarRecords
class CarRecords {
private:
    int arraySize;
    ifstream infile;
    Car *cars;
public:
    //constructor
    CarRecords(int size) {
        //open file
        infile.open("CarRecords.txt");
        //check if the file exist
        if (!infile) {
            cout << "Can't open file CarRecords.txt";
            exit(0);
        }
        //set the number of records to read to 10 if user enter > 10
        if (size > 10) {
            size = 10;
        }
        //initialize the cars pointer to a car array
        cars = new Car[size];
        //set the size of the car array
        arraySize = size;

        //inserting texts to the array, line by line
        for (int i = 0; i < arraySize; i++) {
            string make;
            string model;
            int year;
            string color;

            //get make from file
            infile >> make;
            //delete the last comma from make
            make.erase(make.size() - 1);
            //get model from file
            infile >> model;
            //delete the ending comma from model
            model.erase(model.size() - 1);
            //get year
            infile >> year;
            //get then throw away the comma after the year
            infile >> color;
            //get the actual color from file
            infile >> color;
            //set the fields to the ith element of the array
            (cars + i)->setFields(make, model, year, color);
        }
    }
```

```
//destructor
~CarRecords() {
    //close file
    infile.close();
    //free up cars array memory
    delete [] cars;
}


//print a car object
void printCar(Car *car) {
    cout << car->getMake() << " , "
         << car->getModel() << " , "
         << car->getYear() << " , "
         << car->getColor() << "\n";
}


//print the array of cars
void printCarRecords () {
    int i;
    cout << "\nPRINTING " << arraySize
         << " RECORDS!----------\n";
    //for loop to print all the cars in the array
    for (i = 0; i < arraySize; i++) {
        printCar(cars + i);
    }
}



//sort the cars by make
void sort_cars_by_make() {
    int i, j;
    cout << "\n\nSORT CAR BY MAKE...\n";
    //bubble sort
    for (i = 0; i < arraySize; i++) {
        for (j = 0; j < arraySize - i - 1; j++) {
            //check for make
            if ((cars + j)->getMake() > (cars + j + 1)->getMake()) {
                //make a temp car pointer to swap the 2 car records
                Car *temp = new Car();
                //point temp to the first car
                *temp = *(cars + j);
                //point the first car to the second car
                *(cars + j) = *(cars + j + 1);
                //point the second car to temp
                *(cars + j + 1) = *temp;
            }
        }
    }
}



//sort the cars by year


//Same as above
void sort_cars_by_year() {
    int i, j;
    cout << "\n\nSORT CAR BY YEAR...\n";
    //bubble sort
    for (i = 0; i < arraySize; i++) {
        for (j = 0; j < arraySize - i - 1; j++) {
            //compare 2 years for swapping
            if ((cars + j)->getYear() > (cars + j + 1)->getYear()) {
                //create a temp position
                Car *temp = new Car();
                //point the temp to the first car
                *temp = *(cars + j);
                //point the first to the second
                *(cars + j) = *(cars + j + 1);
                //point the second to temp
                *(cars + j + 1) = *temp;
            }
        }
    }
}



    //print duplicated records
    void print_duplicates() {
        int i, j;
        cout << "\n\nCHECKING FOR DUPLICATES...\n";


        //check each record with the rest
        for (i = 0; i < arraySize; i++) {
            for (j = i + 1; j < arraySize; j++) {
                //check between the 2 records
                if ((cars + i)->getMake() == (cars + j)->getMake()
                        && (cars + i)->getModel() == (cars + j)->getModel()
                        && (cars + i)->getYear() == (cars + j)->getYear()
                        && (cars + i)->getColor() == (cars + j)->getColor()) {
                    //print them both out if similar
                    printCar(cars + i);
                    printCar(cars + j);
                }
            }
        }
        cout << "\nDone\n\n";
    }


};
```

```cpp
//main function
int main() {
    int numRecs;
    cout << "\n\nNumber or Records to read? " ;
    cin >> numRecs;
    CarRecords *cr = new CarRecords(numRecs);



    // Print car records
    cr->printCarRecords();
    // Sort by Year
    cr->sort_cars_by_year();
    // Print car records
    cr->printCarRecords();
    // Sort by Make
    cr->sort_cars_by_make();
    // Print car records
    cr->printCarRecords();
    // Check for Duplicates
    cr->print_duplicates();



    delete cr;
}
```

```
-bash-4.3$ g++ HW1Part3.cpp
-bash-4.3$ ./a.out


Number or Records to read? 5

PRINTING 5 RECORDS!----------------------
Subaru , Outback , 2016 , green
Toyota , Corolla , 2006 , white
Dodge , Neon , 1993 , pink
Ford , Fusion , 2013 , yellow
Honda , Fit , 2015 , blue


SORT CAR BY YEAR...

PRINTING 5 RECORDS!----------------------
Dodge , Neon , 1993 , pink
Toyota , Corolla , 2006 , white
Ford , Fusion , 2013 , yellow
Honda , Fit , 2015 , blue
Subaru , Outback , 2016 , green


SORT CAR BY MAKE...

PRINTING 5 RECORDS!----------------------
Dodge , Neon , 1993 , pink
Ford , Fusion , 2013 , yellow
Honda , Fit , 2015 , blue
Subaru , Outback , 2016 , green
Toyota , Corolla , 2006 , white


CHECKING FOR DUPLICATES...

Done
```

```
Number or Records to read? 10

PRINTING 10 RECORDS!----------------------
Subaru , Outback , 2016 , green
Toyota , Corolla , 2006 , white
Dodge , Neon , 1993 , pink
Ford , Fusion , 2013 , yellow
Honda , Fit , 2015 , blue
Ford , Expedition , 2009 , silver
Toyota , Corolla , 2006 , white
Ford , Fusion , 2013 , yellow
Jeep , Cherokee , 1999 , red
Mazda , Protoge , 1996 , gold


SORT CAR BY YEAR...

PRINTING 10 RECORDS!----------------------
Dodge , Neon , 1993 , pink
Mazda , Protoge , 1996 , gold
Jeep , Cherokee , 1999 , red
Toyota , Corolla , 2006 , white
Toyota , Corolla , 2006 , white
Ford , Expedition , 2009 , silver
Ford , Fusion , 2013 , yellow
Ford , Fusion , 2013 , yellow
Honda , Fit , 2015 , blue
Subaru , Outback , 2016 , green
```

```
SORT CAR BY MAKE...

PRINTING 10 RECORDS!----------------------
Dodge , Neon , 1993 , pink
Ford , Expedition , 2009 , silver
Ford , Fusion , 2013 , yellow
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


CHECKING FOR DUPLICATES...
Ford , Fusion , 2013 , yellow
Ford , Fusion , 2013 , yellow
Toyota , Corolla , 2006 , white
Toyota , Corolla , 2006 , white

Done
```

```
Number or Records to read? 15

PRINTING 10 RECORDS!---------------------
Subaru , Outback , 2016 , green
Toyota , Corolla , 2006 , white
Dodge , Neon , 1993 , pink
Ford , Fusion , 2013 , yellow
Honda , Fit , 2015 , blue
Ford , Expedition , 2009 , silver
Toyota , Corolla , 2006 , white
Ford , Fusion , 2013 , yellow
Jeep , Cherokee , 1999 , red
Mazda , Protoge , 1996 , gold


SORT CAR BY YEAR...

PRINTING 10 RECORDS!---------------------
Dodge , Neon , 1993 , pink
Mazda , Protoge , 1996 , gold
Jeep , Cherokee , 1999 , red
Toyota , Corolla , 2006 , white
Toyota , Corolla , 2006 , white
Ford , Expedition , 2009 , silver
Ford , Fusion , 2013 , yellow
Ford , Fusion , 2013 , yellow
Honda , Fit , 2015 , blue
Subaru , Outback , 2016 , green
```

```
SORT CAR BY MAKE...

PRINTING 10 RECORDS!---------------------
Dodge , Neon , 1993 , pink
Ford , Expedition , 2009 , silver
Ford , Fusion , 2013 , yellow
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


CHECKING FOR DUPLICATES...
Ford , Fusion , 2013 , yellow
Ford , Fusion , 2013 , yellow
Toyota , Corolla , 2006 , white
Toyota , Corolla , 2006 , white

Done
```