Institute of Computer And Technology

B.Tech – CSE(BDA)

<u>Name:-</u> Dwij Vatsal Desai

<u>Sem:-</u> 2

<u>Sub: -</u> ESFP-II

<u>Enrollment No.:-</u> 23162121027

<u>Prac:-</u> 6

<u>Date:-</u> 27/2/2024

<u>Q.1.</u>

Objective: Make sample project using class, object, cout, cin, endl, getline() function, ignore(), and looping.

Preform the following instruction in sample project.

1. Insert minimum 5 newly available cars information in a showroom.

2. Display all newly cars to customer if selected to display.

3. Find most expensive car from the showroom.

4. Find most cheaper car from showroom.

5. Sort the cars by price in ascending or descending order to display as per the customers choice.

<u>Car input:</u>

Car model, Car Brand, Car Manufacturing year, Car Color, Car Price.

<u>Algorithm:-</u>

1. Start

2. Create a structure for entering data about students.

3. Program a code with the use of DMA.

4. Collect the Data from the user.

5. Show the data using printf.

6. Select a person's name.

7. Show the data of the person's name.

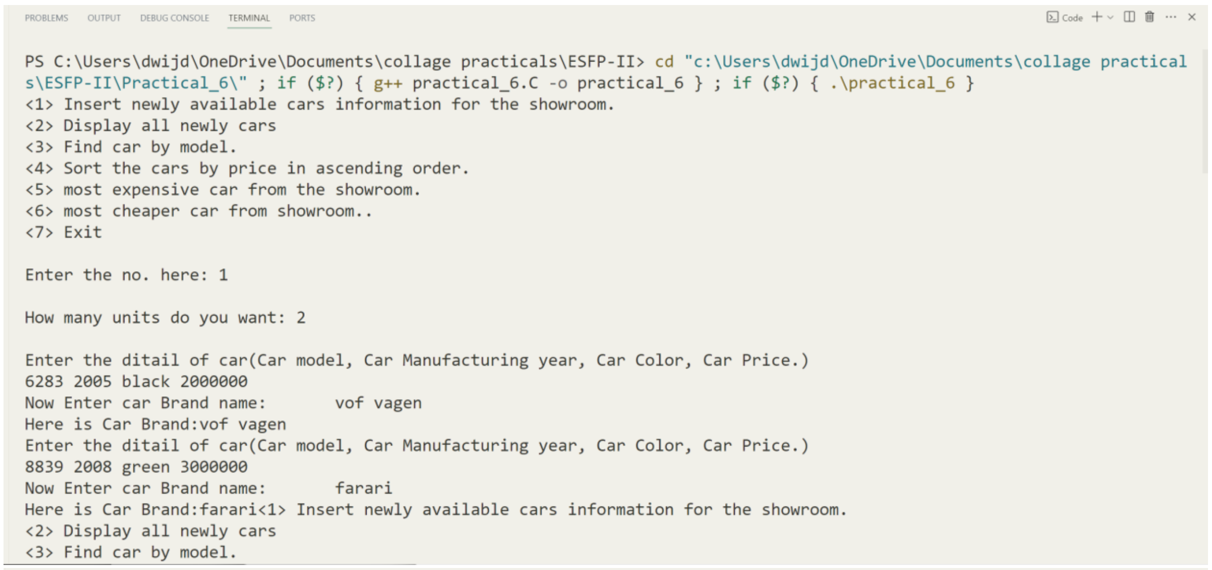
8. End

Code:-

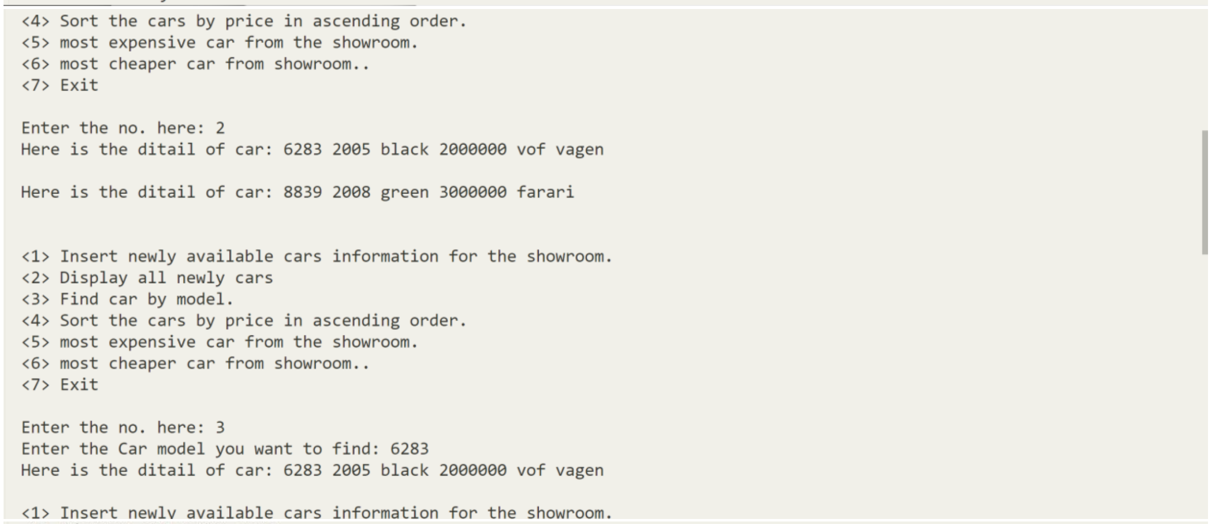
```
/*
Insertion: You'll need a method or function to add new cars to your
showroom.
Think about what information you'll need for each car and how you'll
store this
information in your program.

Display: Consider how you will show the newly available cars to
customers.
Will you print them to the console? Display them in a graphical user
interface
(GUI)? Think about how you want to present this information.

Finding: Think about how you will find the most expensive and
cheapest cars in
your showroom. You'll need to iterate through your list of cars and
compare
their prices to determine which is the most expensive or cheapest.
```

```
Sorting: Consider how you will sort the cars by price in either
ascending or

descending order. You'll need to decide on a sorting algorithm and
implement

it in your program.

*/


#include <iostream>

#include <limits> // for numeric_limits

#include <string>


using namespace std;


class Car

{

   public:

   // Car model, Car Brand, Car Manufacturing year, Car Color, Car
Price.

    string Car_model;

    int Car_year;

    string Car_Color;

    int Car_Price;           // considering Price is in $

    char Car_Brand[20];      // This is for taking getline


    // int found=0;


    Car()

    {}
    void Insertion()

    {

        cout<<endl<<"Enter the ditail of car(Car model, Car 
Manufacturing year, Car Color, Car Price.)\n";

        cin>>Car_model>>Car_year>>Car_Color>>Car_Price;


        cin.ignore();


        cout<<"Now Enter car Brand name:\t";

        cin.getline(Car_Brand,20);


        cout <<"Here is Car Brand:"<<Car_Brand;

    }


    void Display()

    {

        cout <<"Here is the ditail of car:"<<" "<<Car_model<<" 
"<<Car_year<<" "<<Car_Color<<" "<<Car_Price<<" 
"<<Car_Brand<<endl<<endl;

    }


    void Finding(string a)

    {

        int i;


        if(a==Car_model)

        {

            Display();

        }
    }

    static Car* highest(Car cars[], int size)
    {
        if (size <= 0)
        {
            return nullptr; // Or some other value indicating no car found
        }



        Car* mostExpensiveCar = &cars[0]; // Assume the first car is the most expensive
        for (int i = 1; i < size; ++i)
        {
            if (cars[i].Car_Price > mostExpensiveCar->Car_Price)
            {
                mostExpensiveCar = &cars[i]; // Update the most expensive car
            }
        }

        return mostExpensiveCar;
    }

    static Car* lowest(Car cars[], int size)
    {
        if (size <= 0)
        {
            return nullptr; // Or some other value indicating no car
found
        }

        Car* mostAffordableCar = &cars[0]; // Assume the first car
is the most affordable
        for (int i = 1; i < size; ++i)
        {
            if (cars[i].Car_Price < mostAffordableCar->Car_Price)
            {
                mostAffordableCar = &cars[i]; // Update the most
affordable car
            }
        }

        return mostAffordableCar;
    }

    static void sortByPriceAscending(Car cars[], int size)
    {
        for (int i = 0; i < size - 1; ++i)
        {
            for (int j = 0; j < size - i - 1; ++j)
            {
                if (cars[j].Car_Price > cars[j + 1].Car_Price)
                {
                    // Swap cars[j] and cars[j+1]
                    Car temp = cars[j];
                    cars[j] = cars[j + 1];

                    cars[j + 1] = temp;

                }

            }

        }

    }


};


int main()

{

    Car obj[10];

    int num_Car = 0;  // Initialize num_Car to 0


    for(;;)

    {


        int i,Choice;

        string find_car;

        cout<<"<1> Insert newly available cars information for the
showroom."<<endl;

        cout<<"<2> Display all newly cars"<<endl;

        cout<<"<3> Find car by model."<<endl;

        cout<<"<4> Sort the cars by price in ascending
order."<<endl;

        cout<<"<5> most expensive car from the showroom."<<endl;

        cout<<"<6> most cheaper car from showroom.."<<endl;

        cout<<"<7> Exit"<<endl<<endl;

        cout<<"Enter the no. here: ";
        cin>>Choice;

        cin.ignore();

            switch (Choice)
            {
                case 1:
                int numNewCars;

                cout<<"\nHow many units do you want: ";
                cin>>numNewCars;

                for (i = num_Car; i < num_Car + numNewCars && i <
10; i++)
                {
                    obj[i].Insertion();
                }
                num_Car = i;

                break;

                case 2:

                for (i = 0; i < num_Car ; i++)
                {
                    obj[i].Display();
                }

                cout<<endl;
                break;

                case 3:

                cout<<"Enter the Car model you want to find: ";
                cin>>find_car;

                    for (i=0 ; i < num_Car ; i++)
                    {
                        obj[i].Finding(find_car);
                    }

                break;

                case 4:
                    // Sort the cars by price in ascending order
                    Car:: sortByPriceAscending(obj, num_Car);
                    cout << "Cars sorted by price in ascending
order:" << endl;
                    for (i = 0; i < num_Car; ++i)
                    {
                        obj[i].Display();
                    }
                    break;

                case 5:
                {
                    // Find the most expensive car
                    Car* mostExpensiveCar = Car::highest(obj, 
num_Car);
                    if (mostExpensiveCar != nullptr)
                    {
                        cout << "The most expensive car is: " << 
mostExpensiveCar->Car_model << " ($" << mostExpensiveCar->Car_Price 
<< ")" << endl;
                    }
                    else
                    {
                        cout << "No cars found." << endl;
                    }
                    break;
                }


                case 6:
                {
                    // Find the most affordable car
                    Car* mostAffordableCar = Car::lowest(obj, 
num_Car);
                    if (mostAffordableCar != nullptr)
                    {
                        cout << "The most affordable car is: " << 
mostAffordableCar->Car_model << " ($" << mostAffordableCar-
>Car_Price << ")" << endl;
                    }
                    else
                    {
                        cout << "No cars found." << endl;
                    }
                    break;
                }

                case 7:
                    return 0;
                break;

                default:
                cout<<endl<<">>>>Enter right
number<<<<"<<endl<<endl;
                break;
            }
    }

    return 0;
}
```

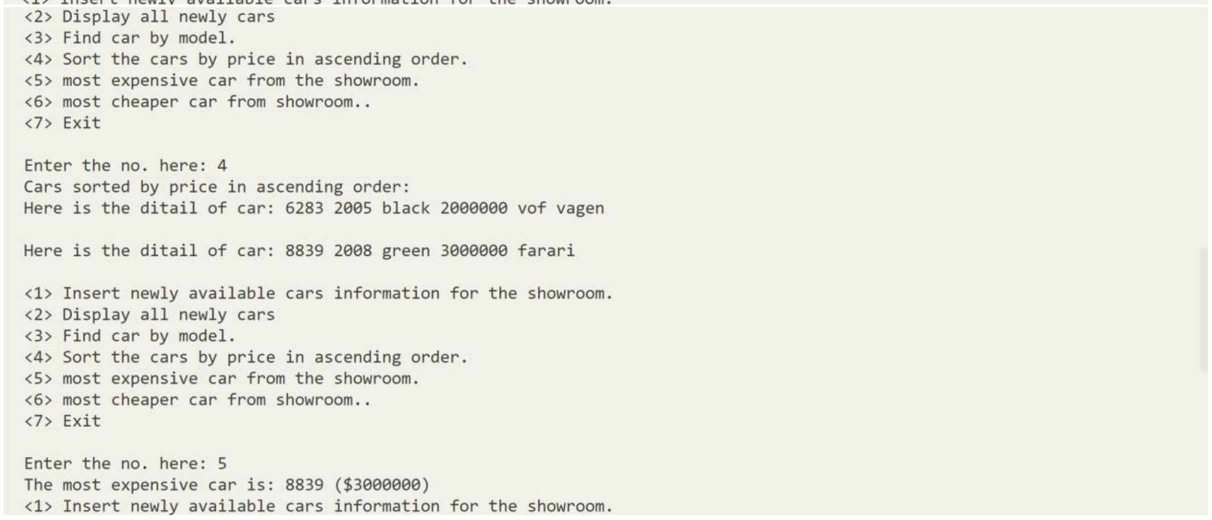
<u>Output-</u>



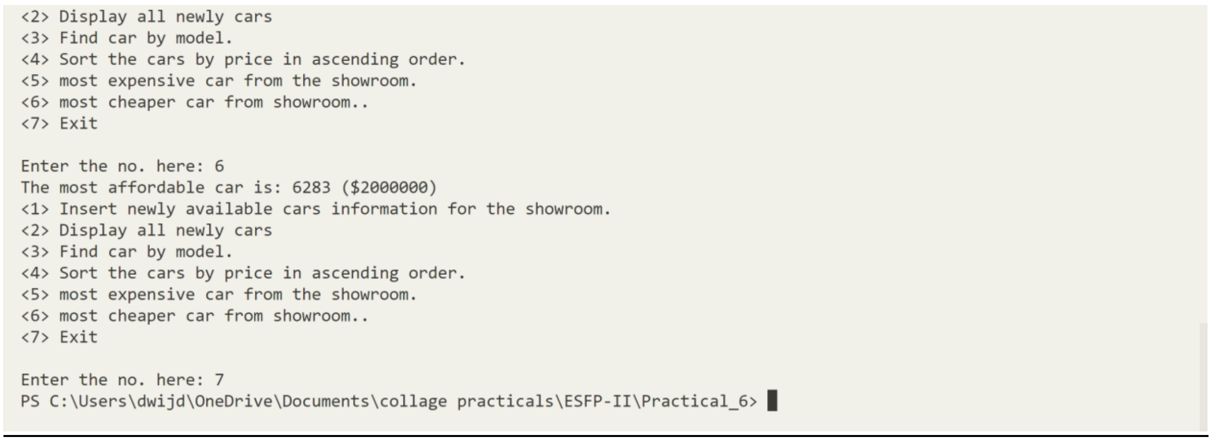
```
PS C:\Users\dwijd\OneDrive\Documents\collage practicals\ESFP-II> cd "c:\Users\dwijd\OneDrive\Documents\collage practical
s\ESFP-II\Practical_6\" ; if ($?) { g++ practical_6.C -o practical_6 } ; if ($?) { .\practical_6 }
<1> Insert newly available cars information for the showroom.
<2> Display all newly cars
<3> Find car by model.
<4> Sort the cars by price in ascending order.
<5> most expensive car from the showroom.
<6> most cheaper car from showroom..
<7> Exit

Enter the no. here: 1

How many units do you want: 2

Enter the ditail of car(Car model, Car Manufacturing year, Car Color, Car Price.)
6283 2005 black 2000000
Now Enter car Brand name:       vof vagen
Here is Car Brand:vof vagen
Enter the ditail of car(Car model, Car Manufacturing year, Car Color, Car Price.)
8839 2008 green 3000000
Now Enter car Brand name:       farari
Here is Car Brand:farari<1> Insert newly available cars information for the showroom.
<2> Display all newly cars
<3> Find car by model.
```



```
<4> Sort the cars by price in ascending order.
<5> most expensive car from the showroom.
<6> most cheaper car from showroom..
<7> Exit

Enter the no. here: 2
Here is the ditail of car: 6283 2005 black 2000000 vof vagen

Here is the ditail of car: 8839 2008 green 3000000 farari


<1> Insert newly available cars information for the showroom.
<2> Display all newly cars
<3> Find car by model.
<4> Sort the cars by price in ascending order.
<5> most expensive car from the showroom.
<6> most cheaper car from showroom..
<7> Exit

Enter the no. here: 3
Enter the Car model you want to find: 6283
Here is the ditail of car: 6283 2005 black 2000000 vof vagen

<1> Insert newly available cars information for the showroom.
```

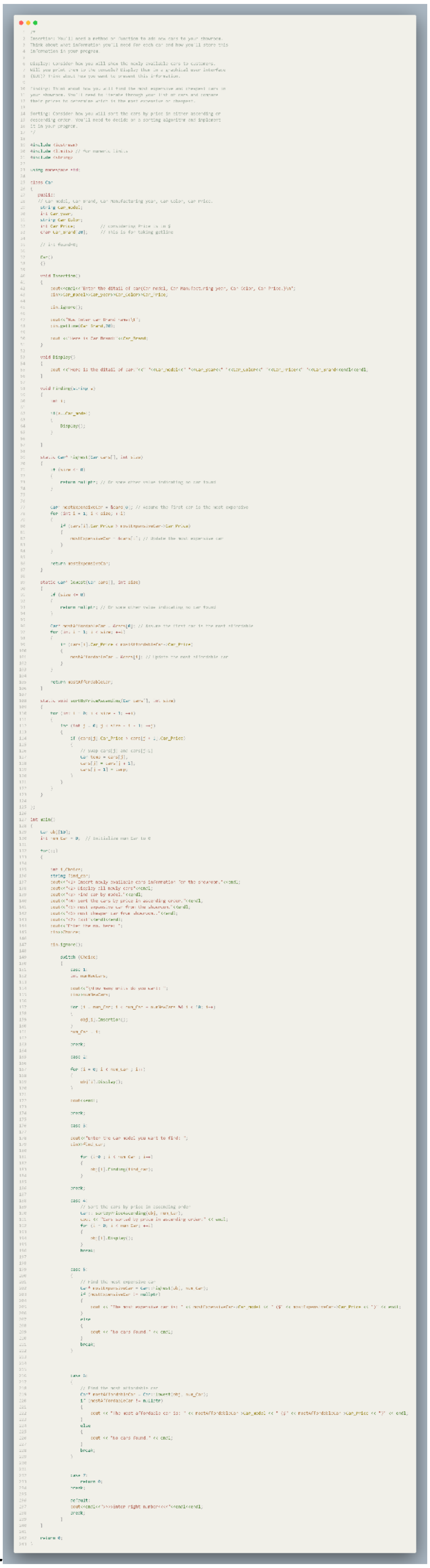


```
<2> Display all newly cars
<3> Find car by model.
<4> Sort the cars by price in ascending order.
<5> most expensive car from the showroom.
<6> most cheaper car from showroom..
<7> Exit

Enter the no. here: 4
Cars sorted by price in ascending order:
Here is the ditail of car: 6283 2005 black 2000000 vof vagen

Here is the ditail of car: 8839 2008 green 3000000 farari

<1> Insert newly available cars information for the showroom.
<2> Display all newly cars
<3> Find car by model.
<4> Sort the cars by price in ascending order.
<5> most expensive car from the showroom.
<6> most cheaper car from showroom..
<7> Exit

Enter the no. here: 5
The most expensive car is: 8839 ($3000000)
<1> Insert newly available cars information for the showroom.
```



```
<2> Display all newly cars
<3> Find car by model.
<4> Sort the cars by price in ascending order.
<5> most expensive car from the showroom.
<6> most cheaper car from showroom..
<7> Exit

Enter the no. here: 6
The most affordable car is: 6283 ($2000000)
<1> Insert newly available cars information for the showroom.
<2> Display all newly cars
<3> Find car by model.
<4> Sort the cars by price in ascending order.
<5> most expensive car from the showroom.
<6> most cheaper car from showroom..
<7> Exit

Enter the no. here: 7
PS C:\Users\dwijd\OneDrive\Documents\collage practicals\ESFP-II\Practical_6>
```

Photo of code:-



```cpp
/*
Insertion: You'll need a method or function to add new cars to your showroom.
Think about what information you'll need for each car and how you'll store this
information in your program.

Display: Consider how you will show the newly available cars to customers.
Will you print them to the console? Display them in a graphical user interface
(GUI)? Think about how you want to present this information.

Finding: Think about how you will find the most expensive and cheapest cars in
your showroom. You'll need to iterate through your list of cars and compare
their prices to determine which is the most expensive or cheapest.

Sorting: Consider how you will sort the cars by price in either ascending or
descending order. You'll need to decide on a sorting algorithm and implement
it in your program.
*/

#include <iostream>
#include <limits> // For numeric limits
#include <string>

using namespace std;

class Car
{
   public:
   // Car model, Car Brand, Car Manufacturing year, Car Color, Car Price.
    string Car_model;
    int Car_year;
    string Car_Color;
    int Car_Price;          // considering Price is in $
    char Car_Brand[20];     // this is for taking getline

    // int found=0;

    Car()
    {}

    void Insertion()
    {
        cout<<endl<<"Enter the ditail of car(Car model, Car Manufacturing year, Car Color, Car Price.)\n";
        cin>>Car_model>>Car_year>>Car_Color>>Car_Price;

        cin.ignore();

        cout<<"Now Enter car Brand name:\t";
        cin.getline(Car_Brand,20);

        cout <<"Here is Car Brand:"<<Car_Brand;
    }

    void Display()
    {
        cout <<"Here is the ditail of car:"<<" "<<Car_model<<" "<<Car_year<<" "<<Car_Color<<" "<<Car_Price<<" "<<Car_Brand<<endl<<endl;
    }

    void Finding(string a)
    {
        int i;

        if(a==Car_model)
        {
            Display();
        }

    }

    static Car* highest(Car cars[], int size)
    {
        if (size <= 0)
        {
            return nullptr; // Or some other value indicating no car found
        }


        Car* mostExpensiveCar = &cars[0]; // Assume the first car is the most expensive
        for (int i = 1; i < size; ++i)
        {
            if (cars[i].Car_Price > mostExpensiveCar->Car_Price)
            {
                mostExpensiveCar = &cars[i]; // Update the most expensive car
            }
        }

        return mostExpensiveCar;
    }

    static Car* lowest(Car cars[], int size)
    {
        if (size <= 0)
        {
            return nullptr; // Or some other value indicating no car found
        }

        Car* mostAffordableCar = &cars[0]; // Assume the first car is the most affordable
        for (int i = 1; i < size; ++i)
        {
            if (cars[i].Car_Price < mostAffordableCar->Car_Price)
            {
                mostAffordableCar = &cars[i]; // Update the most affordable car
            }
        }

        return mostAffordableCar;
    }

    static void sortByPriceAscending(Car cars[], int size)
    {
        for (int i = 0; i < size - 1; ++i)
        {
            for (int j = 0; j < size - i - 1; ++j)
            {
                if (cars[j].Car_Price > cars[j + 1].Car_Price)
                {
                    // Swap cars[j] and cars[j+1]
                    Car temp = cars[j];
                    cars[j] = cars[j + 1];
                    cars[j + 1] = temp;
                }
            }
        }
    }

};

int main()
{
    Car obj[10];
    int num_Car = 0;  // Initialize num_Car to 0

    for(;;)
    {

        int i,Choice;
        string find_car;
        cout<<"<1> Insert newly available cars information for the showroom."<<endl;
        cout<<"<2> Display all newly cars"<<endl;
        cout<<"<3> Find car by model."<<endl;
        cout<<"<4> Sort the cars by price in ascending order."<<endl;
        cout<<"<5> most expensive car from the showroom."<<endl;
        cout<<"<6> most cheaper car from showroom.."<<endl;
        cout<<"<7> Exit"<<endl<<endl;
        cout<<"Enter the no. here: ";
        cin>>Choice;

        cin.ignore();

            switch (Choice)
            {
                case 1:
                int numNewCars;

                cout<<"\nHow many units do you want: ";
                cin>>numNewCars;

                for (i = num_Car; i < num_Car + numNewCars && i < 10; i++)
                {
                    obj[i].Insertion();
                }
                num_Car = i;

                break;

                case 2:

                for (i = 0; i < num_Car ; i++)
                {
                    obj[i].Display();
                }

                cout<<endl;

                break;

                case 3:

                cout<<"Enter the car model you want to find: ";
                cin>>find_car;

                    for (i=0 ; i < num_Car ; i++)
                    {
                        obj[i].Finding(find_car);
                    }

                break;

                case 4:
                    // Sort the cars by price in ascending order
                    Car::sortByPriceAscending(obj, num_Car);
                    cout << "Cars sorted by price in ascending order:" << endl;
                    for (i = 0; i < num_Car; ++i)
                    {
                        obj[i].Display();
                    }
                    break;


                case 5:
                {
                    // Find the most expensive car
                    Car* mostExpensiveCar = Car::highest(obj, num_Car);
                    if (mostExpensiveCar != nullptr)
                    {
                        cout << "The most expensive car is: " << mostExpensiveCar->Car_model << " ($" << mostExpensiveCar->Car_Price << ")" << endl;
                    }
                    else
                    {
                        cout << "No cars found." << endl;
                    }
                    break;
                }



                case 6:
                {
                    // Find the most affordable car
                    Car* mostAffordableCar = Car::lowest(obj, num_Car);
                    if (mostAffordableCar != nullptr)
                    {
                        cout << "The most affordable car is: " << mostAffordableCar->Car_model << " ($" << mostAffordableCar->Car_Price << ")" << endl;
                    }
                    else
                    {
                        cout << "No cars found." << endl;
                    }
                    break;
                }


                case 7:
                    return 0;
                break;

                default:
                cout<<endl<<">>>>Enter right number<<<<"<<endl<<endl;
                break;
            }
    }

    return 0;
}
```