SCSJ 2013-08
DATA STRUCTURE & ALGORITHM

Submitted to: Dr Norsham binti Idris

Submitted by: Meghla Mehnaz Rose A18CS3017
Samiya Hussain Samantha A8CS3030
Khaled Ahmed Sedik Ibrahim A18CS4033
Sarim Ali Ghazi A18CS3031

Team: SSRK

Assignment-2

Topic: Car Rental System

Table of Contents

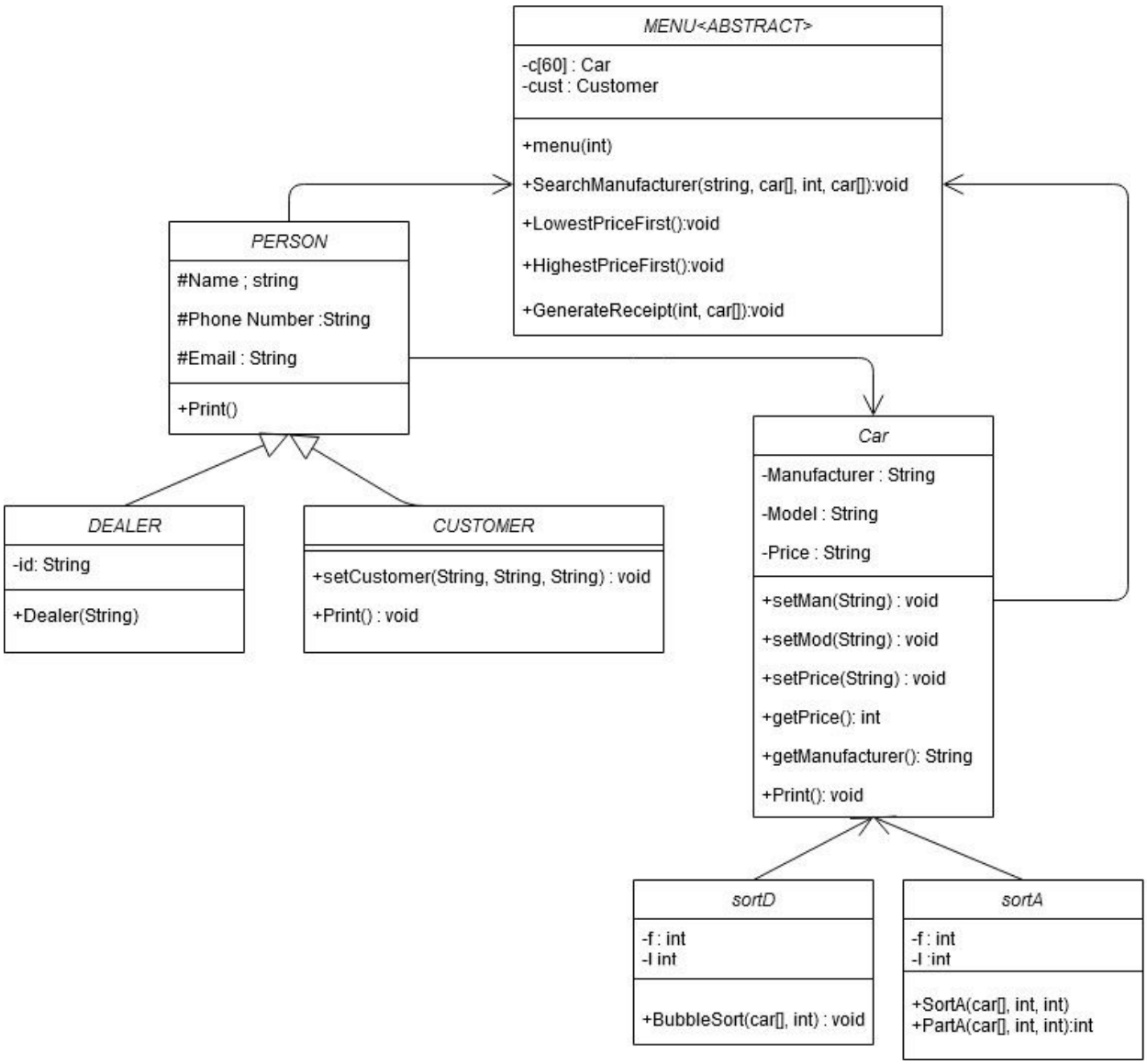
Introduction

Our system is a car rental system, which is mainly used by the customers and the shop-owner. The customer can register or login (if not registered before). Then, he/she can search and select cars for rent. Consequently, making their reservation and paying for it.
The shop-owner maintains car information, views monthly rental information.

Task Allocation

We decided to take a collaborative approach and group programming approach so that we could learn from each other. Hence, all of us did the class diagram together, then divided the classes among us and worked in groups to get the final output. We have collaborated online through Google meet and assisted each other.
It was specially challenging given that all four of us are in different parts of the world and have different time zones. Despite the hardships we are quite satisfied with our output.

Class Diagram



Source Code

```
#include <iostream>
#include <fstream>
#include <string>
```

```
using namespace std;

class Person
{
protected:
    string Name, Email;
    string PhoneNumber;
};

class Dealer : Person
{
    string id;

public:
    Dealer(string num)
    {
        string name, email, phone;
        char read[100];
        ifstream MyFile("dealer.txt");
        while (!MyFile.eof())
        {
            MyFile >> read;
            id = read;

            MyFile >> read;
            Name = read;

            MyFile >> read;
            PhoneNumber = read;

            MyFile >> read;
            Email = read;

            if (id == num)
            {
                cout << "dealer id found " << endl;
                cout << "Dealer Name :" << Name << endl;
                cout << "Dealer  Phone Number :" << PhoneNumber << endl;
                cout << " Dealer Email :" << Email << endl;
                return;
```

```
            }
        }
        cout << "id not found";
    }
};
class Customer : Person
{
public:
    void setCustomer(string n, string p, string e)
    {
        Name = n;
        PhoneNumber = p;
        Email = e;
    }
    void print()
    {
        cout << "customer Name =>" << Name << endl;
        cout << "cutomer Phone =>" << PhoneNumber << endl;
        cout << "customer Email =>" << Email << endl;
    }
};
class car
{
private:
    string Manufacturer, Model, Price;

public:
    car()
    {
        //nothing so far
    }

    void setMan(string s)
    {
        Manufacturer = s;
    }
    void setMod(string s)
    {
        Model = s;
```

```
    }
    void setPrice(string s)
    {
        Price = s;
    }
    int getPrice()
    {
        int x = stoi(Price);

        return x;
    }
    string getManufacturer()
    {
        return Manufacturer;
    }
    void print()
    {
        cout << "Car's Manufacturer =>" << Manufacturer << endl;
        cout << "Car's Model =>" << Model << endl;
        cout << "Car's Price =>" << Price <<"rm"<< endl<<endl;
    }
};

class sortA
{ // using QuickSort Algorithm here
    car a[10];
    int f;
    int l;

public:
    sortA(car a[], int f, int l)
    {
        int cut;
        if (f < l)
        {
            cut = partA(a, f, l);
            sortA(a, f, cut);
            sortA(a, cut + 1, l);
        }
    } // end sortA
```

```
int partA(car a[], int f, int l)
{
    int pivot;
    car temp;
    int loop, cutPoint, bottom, top;
    pivot = a[f].getPrice(); // identify pivot
    bottom = f;
    top = l;
    loop = 1; //always TRUE

    while (loop)
    {

        while (a[top].getPrice() > pivot)
        {
            top--;
        }

        while (a[bottom].getPrice() < pivot)
        {
            bottom++;
        }

        if (bottom < top)
        {
            // change pivot place
            temp= a[bottom];
            a[bottom] = a[top];
            a[top] = temp;
        }
        else
        {
            loop =0; //loop false cutPoint = top:
                cutPoint = top;
        }
    } // end while
    return cutPoint;
} // end partA
```

```
};

class SortD
{

    int f;
    int l;

public:
    void BubbleSort(car a[], int n)
    {
        car temp;
        bool sorted = false; //false when swaps occur
        for (int pass = 1; (pass < n) && !sorted; ++pass)
        {
            //moves the largest element to the end
            sorted = true; //assume sorted
            for (int x = 0; x < n - pass; ++x)
            {
                if (a[x].getPrice() < a[x + 1].getPrice())
                {
                    // exchange items
                    temp = a[x];
                    a[x] = a[x + 1];
                    a[x + 1] = temp;
                    sorted = false; //signal exchange

                } // end if
            }     //end for
        }         //end for
    }
};

class Menu
{
private:
    car c[60];
    Customer cust;

public:
```

```
    Menu(int x)
    {
        char read[50];
        int i = 0;
        ifstream MyFile("car.txt");
        while (!MyFile.eof())
        {
            MyFile >> read;
            c[i].setMan(read);
            MyFile >> read;
            c[i].setMod(read);
            MyFile >> read;
            c[i].setPrice(read);
            cout<<i+1<<".";
            c[i].print();
            i++;
        }
        if (x == 1)
        {
            //customer
            string n, em, p;
            cout << "Hi could we please know your name" << endl;
            cin >> n;
            cout << "Hi could we please know your email" << endl;
            cin >> em;
            cout << "Hi could we please know your Handphone Number" <<
endl;
            cin >> p;
            cust.setCustomer(n, p, em);
            int choice;
            cout << "1. Search for Car via Manufacturer" << endl;
            cout << "2. Search for Cars in Ascending price" << endl;
            cout << "3.search for Cars in Descending Price" << endl;
            cin >> choice;
            if (choice == 1)
            {
                car cManu[60];
                string mans;
                int sel = 0;
```

```
                cout << "What manufacturer are you looking for? !first
letter should be capitall" << endl;
                cin >> mans;
                SearchManufacturer(mans, c, 60,cManu);
                cout << "To Purchase car,enter the number of car on list"
<< endl;
                cout << "To go back, type 0 :";
                cin>>sel;
                if (sel==0)
                {cout<<"going back"<<endl;}
                else
                {GenerateReceipt(sel,cManu);}

            }
            else if (choice == 2)
            {
                LowestPricefirst();

                int choice2;

                cout << "To Purchase car,enter the number of car on list"
<< endl;
                cout << "To go back, type 0 :";
                cin >> choice2;
                if (choice2 == 0)
                {
                    cout << "going back" << endl;
                }
                else
                {
                    GenerateReceipt(choice2, c);
                }
                //do that shis
            }
            else if (choice == 3)
            {
                HighestPricefirst();
                cout << "To Purchase car,enter the number of car on list"
<< endl;
                cout << "To go back, type 0 :";
```

```
                int choice2 = 0;
                 cin >> choice2;
                 if (choice2 == 0)
                 {
                     cout << "going back" << endl;
                 }
                 else
                 {
                     GenerateReceipt(choice2, c);
                 }
             }
             else
             {
                 cout << "wrong input" << endl;
             }
         }
         else if (x == 2)
         {
             //dealer
             string ID;
             cout << "enter id" << endl;
             cin >> ID;
             Dealer d(ID);
         }
         else
         {
             cout << "wrong input";
         }
     }
      void  SearchManufacturer(string search_key, car array[], int
array_size, car newarray[])
     {

         int p;
         int i = 0;
         int index = -1;
         //-1 means record is not found
         for (p = 0; p < array_size; p++)
             for (p = 0; p < array_size; p++)
             {
```

```
                if (search_key == array[p].getManufacturer())
                {
                    index = p;
                    newarray[i] = array[p];
                    cout << i + 1 <<" . ";
                    newarray[i].print();
                    i++;
                }

            } //end for

        // return location of value
    } //end function

    void LowestPricefirst()
    {
       sortA sort1(c,0,59);
       for(int x=0; x<60;x++)
       {
           cout << x+1 << ".";c[x].print();
       }

    }
    void HighestPricefirst()
    {
       SortD sort2;
       sort2.BubbleSort(c,60);
       for (int x = 0; x < 60; x++)
       {
          cout<<x+1<<"."; c[x].print();
       }
    }
    void GenerateReceipt(int i, car s[])
    {
        cust.print();
        s[i - 1].print();
        cout << "congrats on your new car buddy" << endl;
    }
};
```

```
int main()
{
    int person;
    string ID;
    cout << "1. for Customer, 2. Dealer";

    cin >> person;
    Menu m1(person);

    int x;
    cin >> x;
}
```

Searching and Sorting Technique Used

We used bubble sort to sort the prices. This is due to its simplicity and also because our data set is small. It is simple to write, easy to understand and it only takes a few lines of code. The data is sorted in place so there is little memory overhead and, once sorted, the data is in memory, ready for processing. For the best case scenario, there will be no sorting.
Sequential search usually is implemented to search items from an unsorted list/ array. We use sequential search here as our data set is small and the technique can be implemented on a small size of list. Furthermore, it is used for searching that involves records stored in the main memory (RAM).
The searching strategy is mentioned below:
•Examines each element in the array one by one (sequentially) and compares its value with the one being looked for – the search key.

•Search is successful if the search key matches with the value being compared in the array.

Searching process is terminated.

•Else, if no matches are found, the search process is continued to the last element of the array.

•Search is failed if there are no matches found from the array.

Dataset

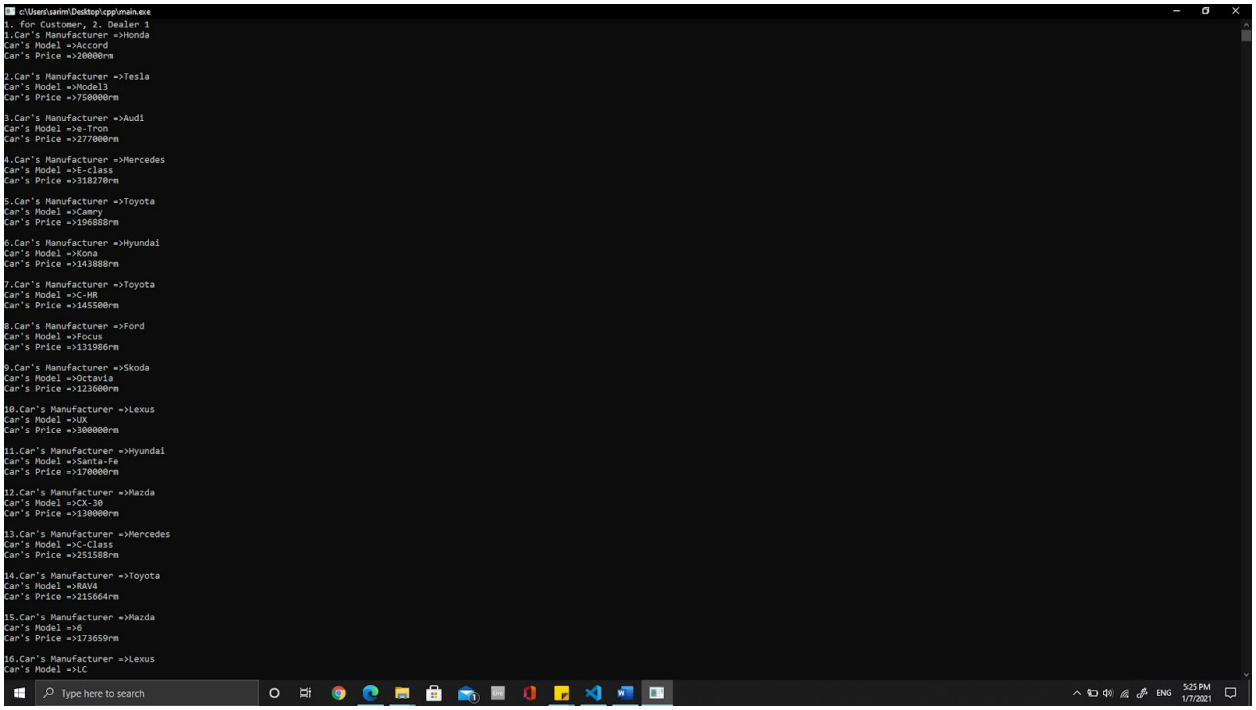
Below are our dummy data set:
Car
Honda Accord 20000
Tesla Model3 750000
Audi e-Tron 277000
Mercedes E-class 318270
Toyota Camry 196888
Hyundai Kona 143888
Toyota C-HR 145500
Ford Focus 131986
Skoda Octavia 123600
Lexus UX 300000
Hyundai Santa-Fe 170000
Mazda CX-30 130000
Mercedes C-Class 251588
Toyota RAV4 215664
Mazda 6 173659
Lexus LC 1250000
Mercedes S-Class 650112
Ford Fiesta 88388
Volkswagen Golf 156000
Honda Civic 114000
Audi Q8 727900
Hyundai i30 298888
Hyundai Tucson 124000
Volkswagen Polo 77646
Suzuki Swift 102888
Volkswagen Passat 195390
Volvo S60 295888
BMW i8 1508800
BMW i3 279000
Toyota Prius 175000
Ford Mustang 598888
Nissan Leaf 189000
Hyundai i10 52188
Hyundai Ioniq 103888
Honda CR-V 140000
Toyota Land-Cruiser 139888
Volkswagen Tiguan 209575
Audi A5 377900
Mercedes GLC 396869
Tesla Model-S 719888

Mazda MX-5 176800
Nissan Qashqai 134888
Volvo V90 388888
Lexus ES 332888
Mercedes GLE 584720
Ford Mondeo 189086
Audi A7 610000
Volvo XC90 413888
Volkswagen T-Cross 123888
BMW X3 314000
Toyota Supra 568000
BMW X5 441000
Mercedes A-Class 222120
Audi Q5 340000
Bentley Continental-GT 868800
Ford S-Max 221728
Volvo XC60 343888
Toyota Tundra 33675
Jaguar XF 460000
Jaguar I-Pace 1439766

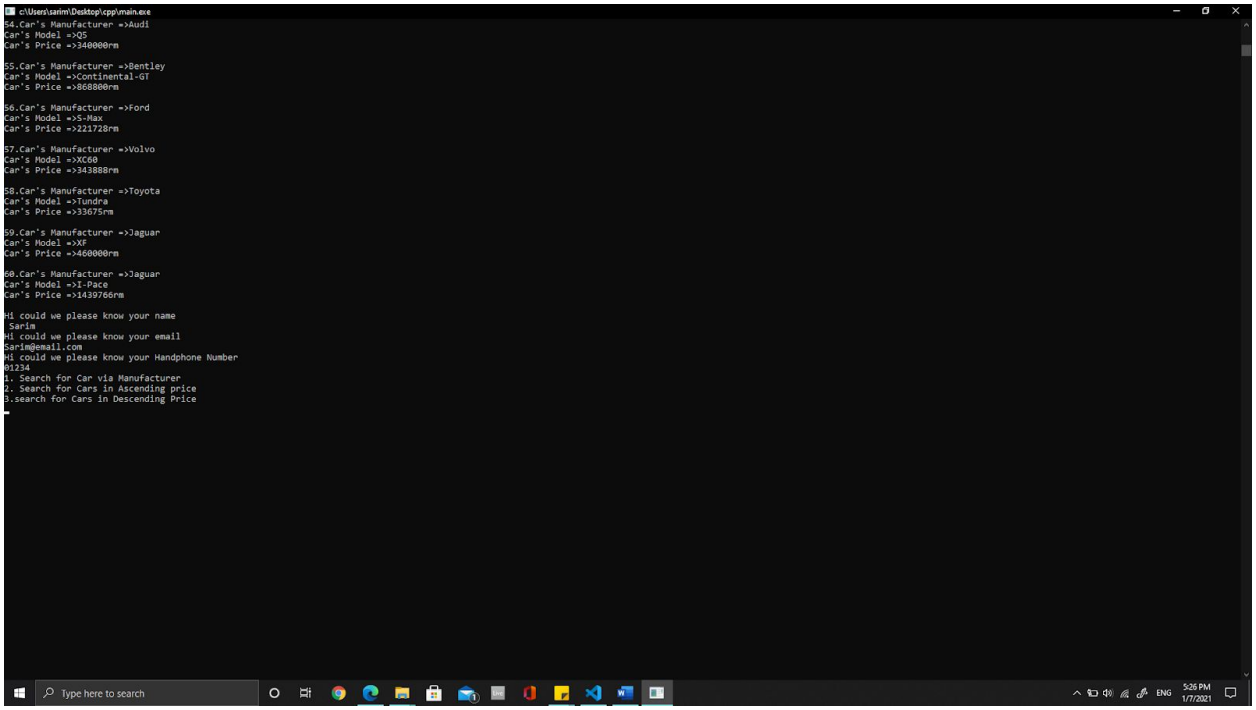
Dealer
1234 Carlos 01111111111 Carloser@gmail.com
4321 Benjamin 01222222222 benjaminbutton@gmail.com
3412 Greta 0133333333 Gretagurl@gmail.com
2143 Mohito 01444444444 binmohitbun@gmail.com

Output

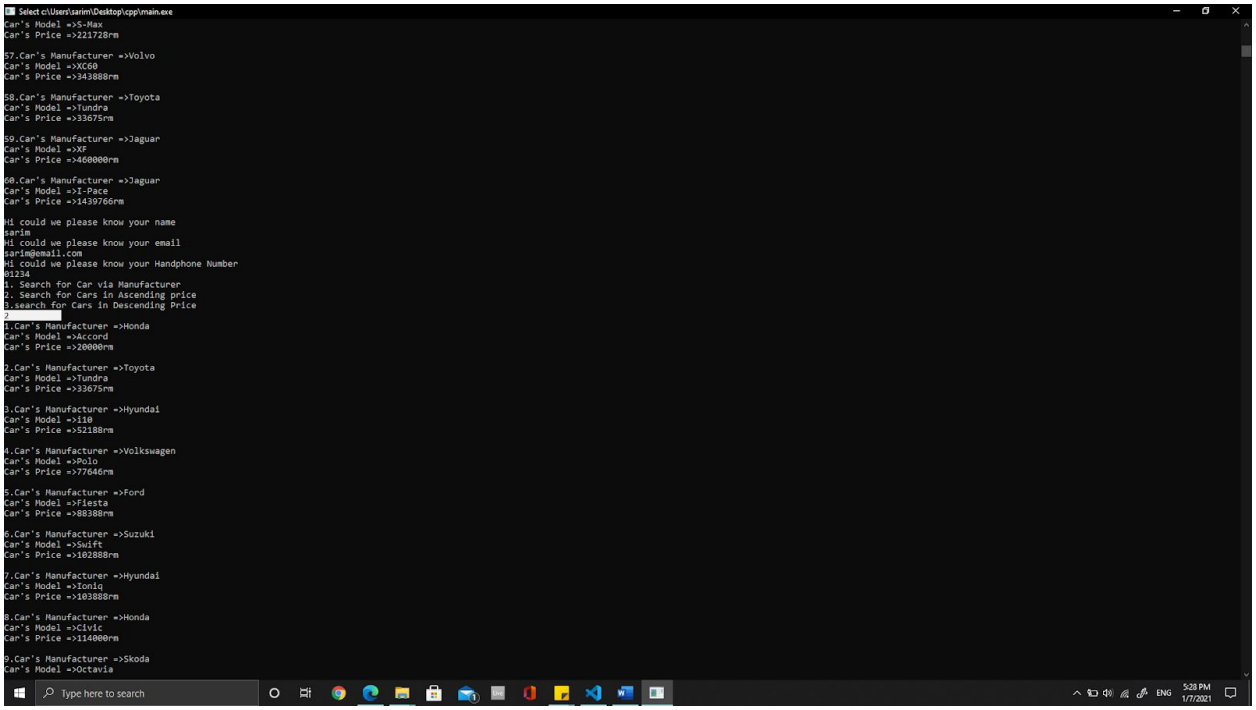
The output produced is shown in the screenshots taken after we ran the program
Customer 1


c:\Users\sarim\Desktop\cpp\main.exe
1. for Customer, 2. Dealer 1
1.Car's Manufacturer =>Honda
Car's Model =>Accord
Car's Price =>200000rm
2.Car's Manufacturer =>Tesla
Car's Model =>Model3
Car's Price =>750000rm
3.Car's Manufacturer =>Audi
Car's Model =>e-Tron
Car's Price =>277000rm
4.Car's Manufacturer =>Mercedes
Car's Model =>E-class
Car's Price =>318270rm
5.Car's Manufacturer =>Toyota
Car's Model =>Camry
Car's Price =>196888rm
6.Car's Manufacturer =>Hyundai
Car's Model =>Kona
Car's Price =>143888rm
7.Car's Manufacturer =>Toyota
Car's Model =>C-HR
Car's Price =>145500rm
8.Car's Manufacturer =>Ford
Car's Model =>Focus
Car's Price =>131986rm
9.Car's Manufacturer =>Skoda
Car's Model =>Octavia
Car's Price =>123600rm
10.Car's Manufacturer =>Lexus
Car's Model =>UX
Car's Price =>300000rm
11.Car's Manufacturer =>Hyundai
Car's Model =>Santa-Fe
Car's Price =>170000rm
12.Car's Manufacturer =>Mazda
Car's Model =>CX-30
Car's Price =>130000rm
13.Car's Manufacturer =>Mercedes
Car's Model =>C-Class
Car's Price =>251588rm
14.Car's Manufacturer =>Toyota
Car's Model =>RAV4
Car's Price =>215664rm
15.Car's Manufacturer =>Mazda
Car's Model =>6
Car's Price =>173659rm
16.Car's Manufacturer =>Lexus
Car's Model =>LC
Type here to search
ENG
5:25 PM
1/7/2021

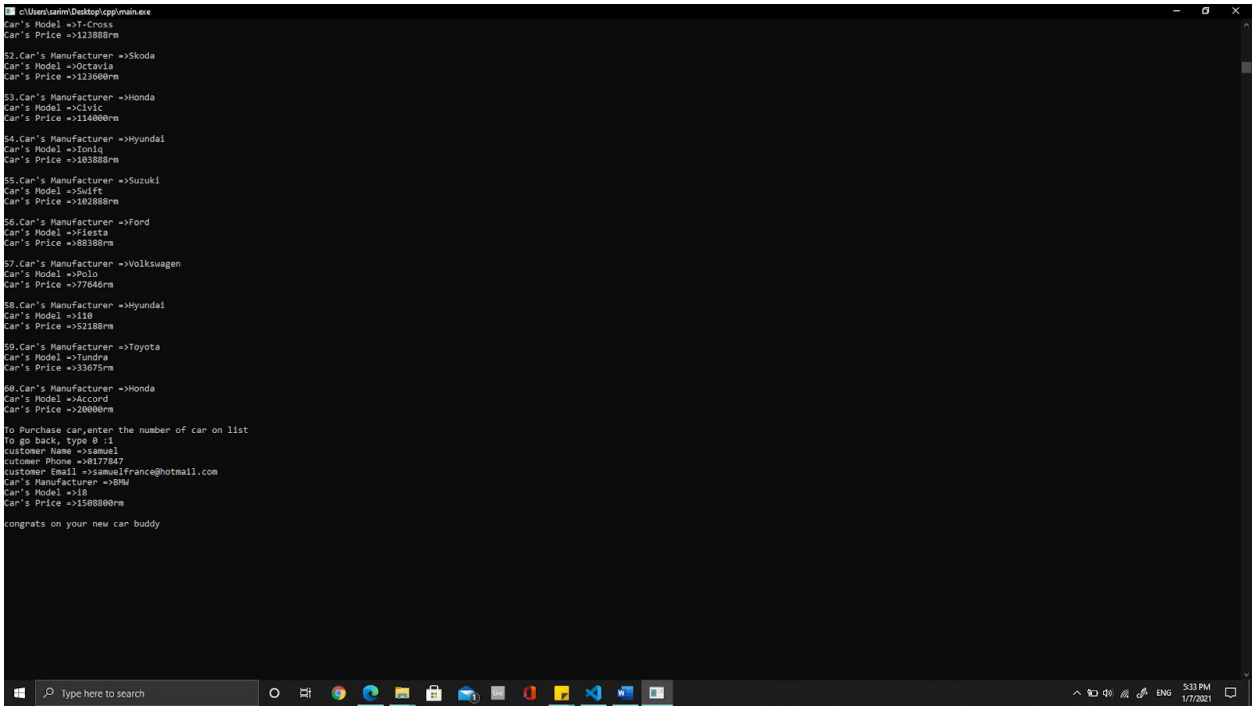
Customer 2



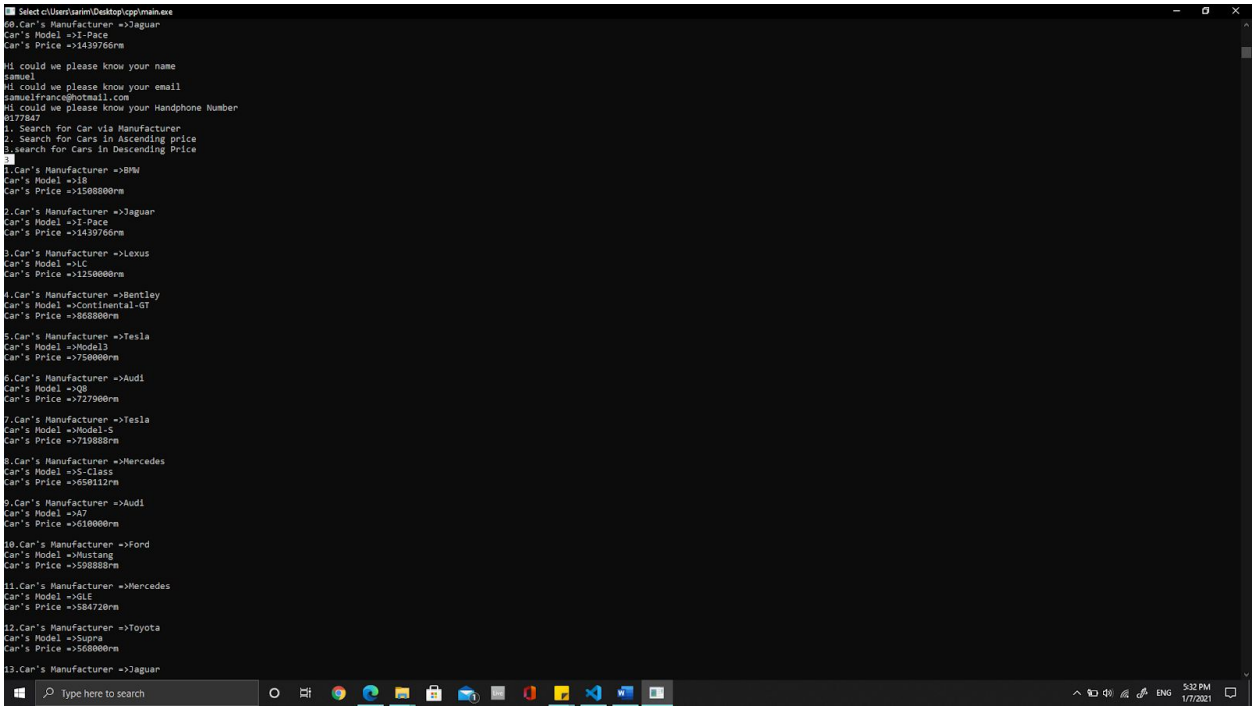
Customer ascending



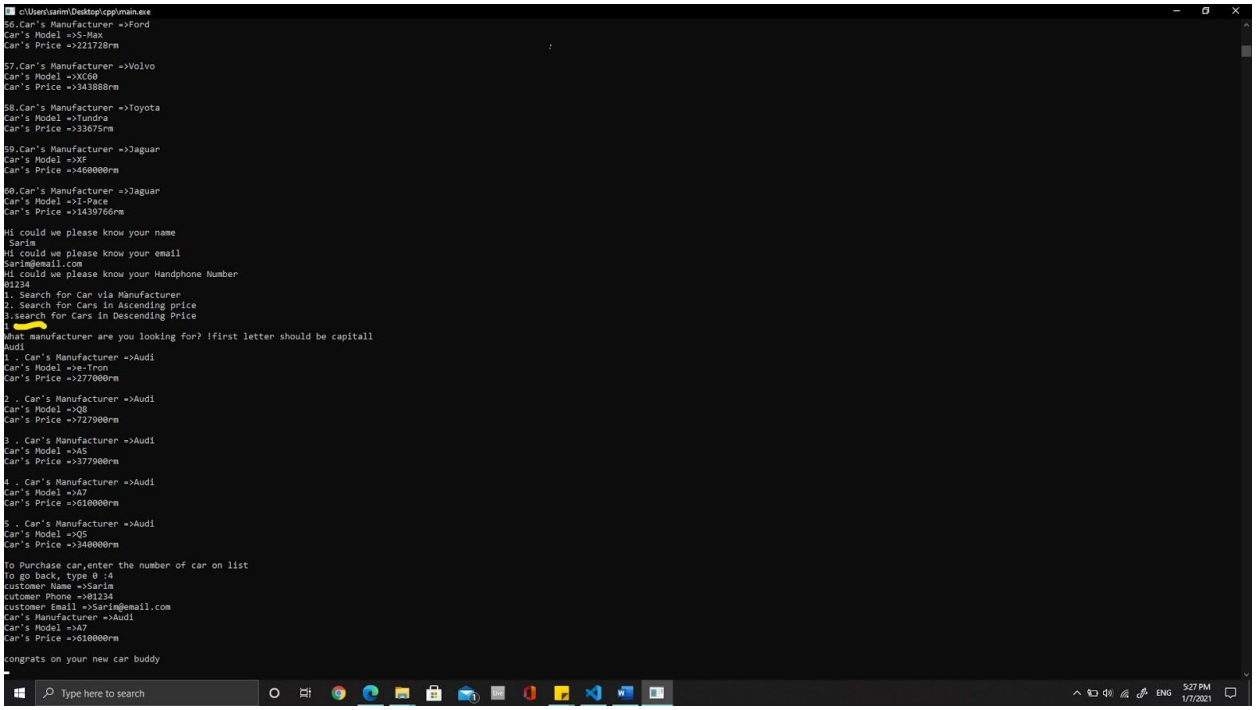
Customer descending output



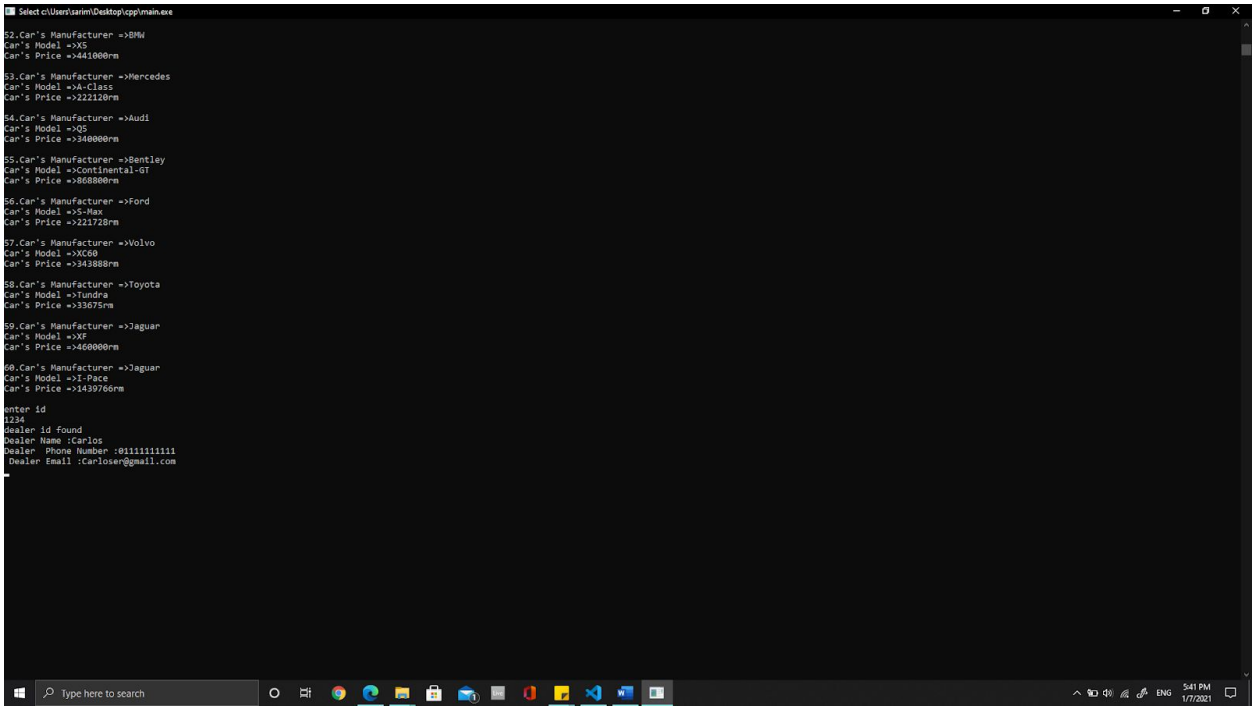
Customer descending



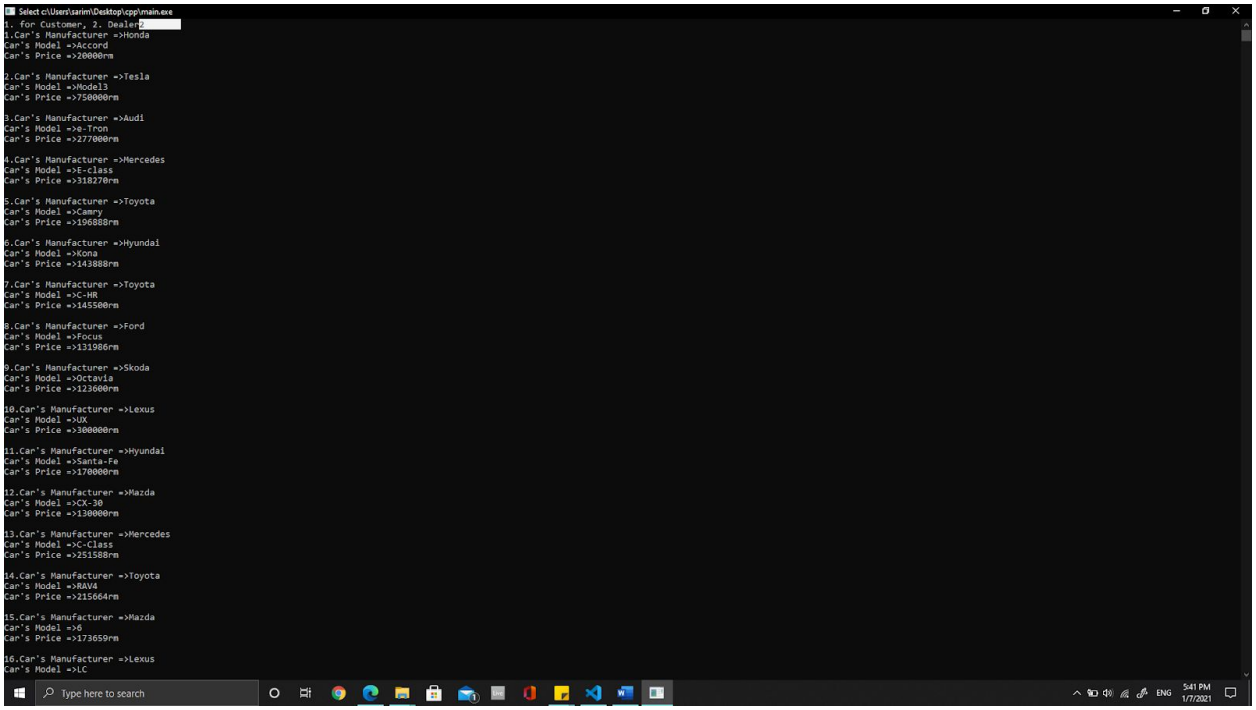
Customer Sequential Search



Dealer FInal



Dealer


Select c:\Users\sarim\Desktop\cpp\main.exe
1. for Customer, 2. Dealer:
1.Car's Manufacturer =>Honda
Car's Model =>Accord
Car's Price =>20000rm
2.Car's Manufacturer =>Tesla
Car's Model =>Model3
Car's Price =>750000rm
3.Car's Manufacturer =>Audi
Car's Model =>e-Tron
Car's Price =>277000rm
4.Car's Manufacturer =>Mercedes
Car's Model =>E-class
Car's Price =>318270rm
5.Car's Manufacturer =>Toyota
Car's Model =>Camry
Car's Price =>196888rm
6.Car's Manufacturer =>Hyundai
Car's Model =>Kona
Car's Price =>143888rm
7.Car's Manufacturer =>Toyota
Car's Model =>C-HR
Car's Price =>145500rm
8.Car's Manufacturer =>Ford
Car's Model =>Focus
Car's Price =>131986rm
9.Car's Manufacturer =>Skoda
Car's Model =>Octavia
Car's Price =>123600rm
10.Car's Manufacturer =>Lexus
Car's Model =>UX
Car's Price =>300000rm
11.Car's Manufacturer =>Hyundai
Car's Model =>Santa-Fe
Car's Price =>170000rm
12.Car's Manufacturer =>Mazda
Car's Model =>CX-30
Car's Price =>130000rm
13.Car's Manufacturer =>Mercedes
Car's Model =>C-Class
Car's Price =>251588rm
14.Car's Manufacturer =>Toyota
Car's Model =>RAV4
Car's Price =>215664rm
15.Car's Manufacturer =>Mazda
Car's Model =>6
Car's Price =>173659rm
16.Car's Manufacturer =>Lexus
Car's Model =>LC
Type here to search
ENG
5:41 PM
1/7/2021