**Lab 5**

**Name: \_\_\_Jorge A. Serrano\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID#: \_121260\_\_\_\_\_\_\_\_\_\_\_\_**

1. Copy the source code developed for Lab 5 and paste it as **text** below. (*15 points*)

Dealer.h

#pragma once

/\*

\* CECS 2223, Computer Programming II Lab

\* Fall 2025, Sec. 05

\* Date: October 14, 2022

\* Topic: Lab 5 - Composition

\* File name: Dealer.h

\* This file declares a class named Dealer

\* Name: Jorge A. Serrano, ID#121260

\*/

// What to include???

#include<iostream>

#include "Cars.h"

using namespace std;

class Dealer {

private:

int carCount; // total cars in inventory, i.e., the size of the array

Cars\*\* inventory; // array to store the dealer's car inventory

string name; // car dealer's name

public:

Dealer(); // the default constructor

Dealer(string); // parameterized constructor, includes car dealer's name

void setName(string); // set the car dealer's name

void addCarToInventory(string, string); // adds a car to inventory

void sellCar(string, string); // sell a car from inventory

int findCar(string, string) const; // searches for a car in inventory

string getName() const; // to get the dealer's name

int getCarCount() const; // gets the total number of cars in inventory

void printInventory() const; // prints a list of each car in inventory, including the serial number

void orderInventory(); // orders the cars in inventory alphabetically

};

Dealer.cpp

/\*

\* CECS 2223, Computer Programming II Lab

\* Fall 2022, Sec. 05

\* Date: October 14, 2022

\* Topic: Lab 5 - Composition

\* File name: Dealer.cpp

\* This file defines the Dealer class

\* Name: Jorge A. Serrano, ID#121260

\*/

#include"Dealer.h"

// The default constructor initializes carCount to 0, inventory to the null pointer (nullptr)

// and name to the empty string ""

Dealer::Dealer()

{

carCount = 0;

inventory = nullptr;

name = " ";

}

// The parameterized constructor receives the string with the dealer's name as parameter,

// initializes carCount to 0, and inventory to the null pointer

Dealer::Dealer(string dealerName)

{

carCount = 0;

inventory = nullptr;

name = dealerName;

}

// The setName method recieves the string with the dealer's name as parameter

void Dealer::setName(string dealerName)

{

name = dealerName;

}

// addCarToInventory receives a car's brand and model as parameters, and adds it to

// the inventory array as a pointer to a Cars object. This method increments carCount

// by 1. Once the car has been added to inventory, the method orderInventory

// must be called to order the array.

void Dealer::addCarToInventory(string carBrand, string carModel)

{

if (getCarCount()>0)

{

cout << "Car list created\n";

Cars\*\* temp = inventory;

inventory = new Cars \* [++carCount];

for (int i = 0; i < carCount; i++)

{

inventory[i] = temp[i];

}

delete[] temp;

inventory[getCarCount() - 1] = new Cars;

inventory[getCarCount() - 1]->setBrand(carBrand);

inventory[getCarCount() - 1]->setModel(carModel);

}

else if (getCarCount() == 0)

{

cout << "Car list created\n";

inventory = new Cars \* [++carCount];

inventory[0] = new Cars;

inventory[getCarCount() - 1] = new Cars;

inventory[getCarCount() - 1]->setBrand(carBrand);

inventory[getCarCount() - 1]->setModel(carModel);

}

else {

cout << "\n\tERROR! The array's size MUST be greater than 0\n\n";

}

orderInventory();

}

// The sellCar method deletes a Cars object from inventory. It must first make sure that

// the car exists in inventory by calling the findCar method. If the car is not found,

// print the message "The selected car, [brand] - [model], was not found in inventory"

// If the car is found, create a new array with one less position, and copy the

// contents of the old array into the new one, except for the car being sold. Once all

// remaining pointers have been copied to the new array, make usre to call the Cars

// destructor to remove the Cars object. This method decrements carCount by 1.

// Once the car has been removed from inventory, the method orderInventory

// must be called to order the array.

void Dealer::sellCar(string carBrand, string carModel)

{

const int carIndex = findCar(carBrand, carModel);

if (getCarCount() > 0)

{

if (carIndex != -1)

{

Cars\*\* temp = inventory;

inventory = new Cars \* [--carCount];

int currentIndex = 0;

for (int i = 0; i < carCount + 1; i++)

{

if (i != carIndex && currentIndex < carCount)

{

inventory[currentIndex] = temp[i];

currentIndex++;

}

}

temp[carIndex]->~Cars();

delete[] temp;

}

else

cout << "\nThe selected car, " << carBrand << " - " << carModel << ", was not found in inventory\n\n";

}

else {

}

orderInventory();

}

// This method searches inventory to find a match for the given car.

// If found, the method returns the index of the car in the array,

// otherwise, it returns -1.

int Dealer::findCar(string carBrand, string carModel) const

{

if (getCarCount() > 0)

{

for (int i = 0; i < carCount; i++)

{

if (inventory[i]->getBrandName() == carBrand && inventory[i]->getModelName() == carModel)

{

return i;

}

}

return -1;

}

else

return -1;

}

// The getter for dealer's name

string Dealer::getName() const

{

return name;

}

// getCarCount returns the total number of cars in inventory

int Dealer::getCarCount() const

{

return carCount;

}

// This method prints a list of all cars in inventory in a table format.

// The method must print the phrase "Dealer [name] has [carCount] cars in inventory."

// Use the strings "BRAND", "MODEL", and "SERIAL NUMBER" as the table's header.

// Make sure to include code for the case where there are no cars in inventory

void Dealer::printInventory() const

{

cout << "Dealer " << name << " has " << carCount << " cars in inventory."<<endl;

cout << "BRAND\tMODEL\tSERIAL NUMBER" << endl;

for (int i = 0; i < carCount; i++)

{

inventory[i]->printCar();

}

cout << endl;

}

// This methods orders the cars in inventory alphabetically, and has no

// parameters. Implement a selection sort algorithm, where the first item

// to be ordered is the brand name, followed by the model name.

// Recall that inventory ordering is only required if there is more than

// one car in inventory

void Dealer::orderInventory()

{

if (getCarCount() > 1)

{

bool swap=0;

do

{

for (int i=0; i < (getCarCount() - 1); i++)

{

swap = false;

if (inventory[i]->getBrandName() > inventory[(i + 1)]->getBrandName())

{

swap = true;

Cars tmp{};

tmp.setBrand(inventory[i]->getBrandName());

tmp.setModel(inventory[i]->getModelName());

inventory[i]->setBrand(inventory[(i + 1)]->getBrandName());

inventory[i]->setModel(inventory[(i + 1)]->getModelName());

inventory[(i + 1)]->setBrand(tmp.getBrandName());

inventory[(i + 1)]->setModel(tmp.getModelName());

}

}

} while (swap);

}

}

lab05.cpp

/\*

\* CECS 2223, Computer Programming II Lab

\* Fall 2022, Sec. 05

\* Date: October 14, 2022

\* Topic: Lab 5 - Composition

\* File name: lab05.cpp

\* This file implements the Dealer class

\* Name: Jorge A. Serrano, ID#121260

\*/

#include "Dealer.h"

#include"Cars.h"

#include <string>

using namespace std;

int menu(); // options menu

bool execute(int, Dealer&); // implements the action from the menu

int main() {

// declare a Dealer object named poli using the parameterized constructor

// with the string "Poli Auto Sales" as argument

Dealer poli ("Poli Auto Sales");

// Develop a while iteration control structure which uses the retun value of

// the execute function as sentinel. This structure has only one line of

// code, the call to execute with menu as argument.

// During program execution add at least six cars to inventory, making sure

// to add at least two cars of the same brand and model, and two cars of the

// same brand but different models. You should also sell at least three cars.

while (execute(menu(), poli) == true);

// write a statement which prints the phrase

// "Program developed by [YOUR NAME], ID#[YOUR ID NUMBER]"

// where the square brackets and the text within is substituted with

// your personal information.

cout << "Program developed by Jorge A. Serrano, ID#121260";

system("pause"); // the pause statement is optional

return 0; // the return statement is required

}

// The menu method presents a list of options to the user.

// It has been defined for you

int menu() {

int option = 0;

cout << "Select one of the following options:\n";

cout << "\t1. Add a car to inventory\n";

cout << "\t2. Sell a car from inventory\n";

cout << "\t3. Print the inventory\n";

cout << "\t4. Exit the program\n";

cout << "\n\tYour selection: ";

cin >> option;

cout << endl;

cin.ignore(256, '\n'); // to "clear" the stdin buffer

return option;

}

// The execute method implements the action selected by the user and receives

// the option number and the address of a Dealer object as parameters.

// Implement the logic using a switch-case structure where each case value

// is that of a valid option number. Make sure to include the default case!

// The method returns false only if the option selected is option 4,

// otherwise, it returns true.

bool execute(int option, Dealer& aDealer) {

string brand = "", model = "";

switch (option) {

case 1:

cout << "Enter the car's brand name: ";

getline(cin, brand, '\n');

cout << "Enter the car's model name: ";

getline(cin, model, '\n');

aDealer.addCarToInventory(brand, model);

cout << endl;

return true;

case 2:

cout << "Enter the car's brand name: ";

getline(cin, brand, '\n');

cout << "Enter the car's model name: ";

getline(cin, model, '\n');

aDealer.sellCar(brand, model);

cout << endl;

return true;

case 3:

aDealer.printInventory();

return true;

case 4:

cout << "Thank you for using this program!\n\n";

return false;

default:

cout << "\n\tERROR! Option #" << option << " is not valid.\n\tTry again!\n\n";

return true;

}

}

1. Paste the screenshots of the program’s execution below. (*5 points*)

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 1

Enter the car's brand name: Toyota

Enter the car's model name: Yaris

Car list created

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 1

Enter the car's brand name: Toyota

Enter the car's model name: Corolla

Car list created

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 1

Enter the car's brand name: Ford

Enter the car's model name: Mustang

Car list created

Destructor

Destructor

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 1

Enter the car's brand name: Jeep

Enter the car's model name: Cherokee

Car list created

Destructor

Destructor

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 1

Enter the car's brand name: Dodge

Enter the car's model name: FIrebird

Car list created

Destructor

Destructor

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 1

Enter the car's brand name: Mitsubishi

Enter the car's model name: Mirage

Car list created

Destructor

Destructor

Destructor

Destructor

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 3

Dealer Poli Auto Sales has 6 cars in inventory.

BRAND MODEL SERIAL NUMBER

Dodge FIrebird TOYYAR002

Ford Mustang TOYCOR003

Jeep Cherokee FORMUS004

Mitsubishi Mirage JEECHE007

Toyota Yaris DODFIR010

Toyota Corolla MITMIR013

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 2

Enter the car's brand name: Ford

Enter the car's model name: Mustang

Destructor

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 2

Enter the car's brand name: Toyota

Enter the car's model name: Yaris

Destructor

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 2

Enter the car's brand name: Dodge

Enter the car's model name: FIrebird

Destructor

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

3. Print the inventory

4. Exit the program

Your selection: 3

Dealer Poli Auto Sales has 3 cars in inventory.

BRAND MODEL SERIAL NUMBER

Jeep Cherokee FORMUS004

Mitsubishi Mirage JEECHE007

Toyota Corolla MITMIR013

Select one of the following options:

1. Add a car to inventory

2. Sell a car from inventory

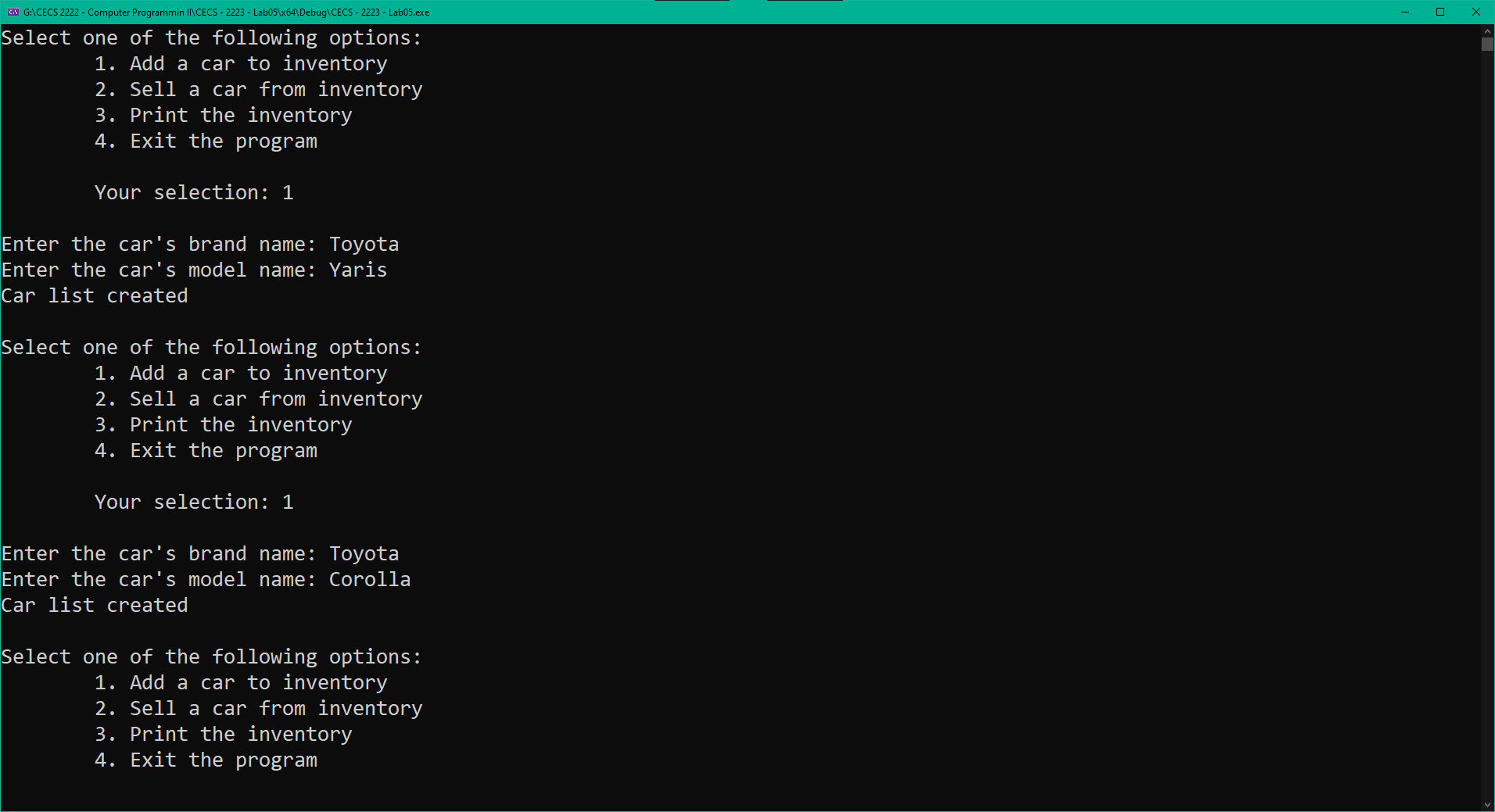
3. Print the inventory

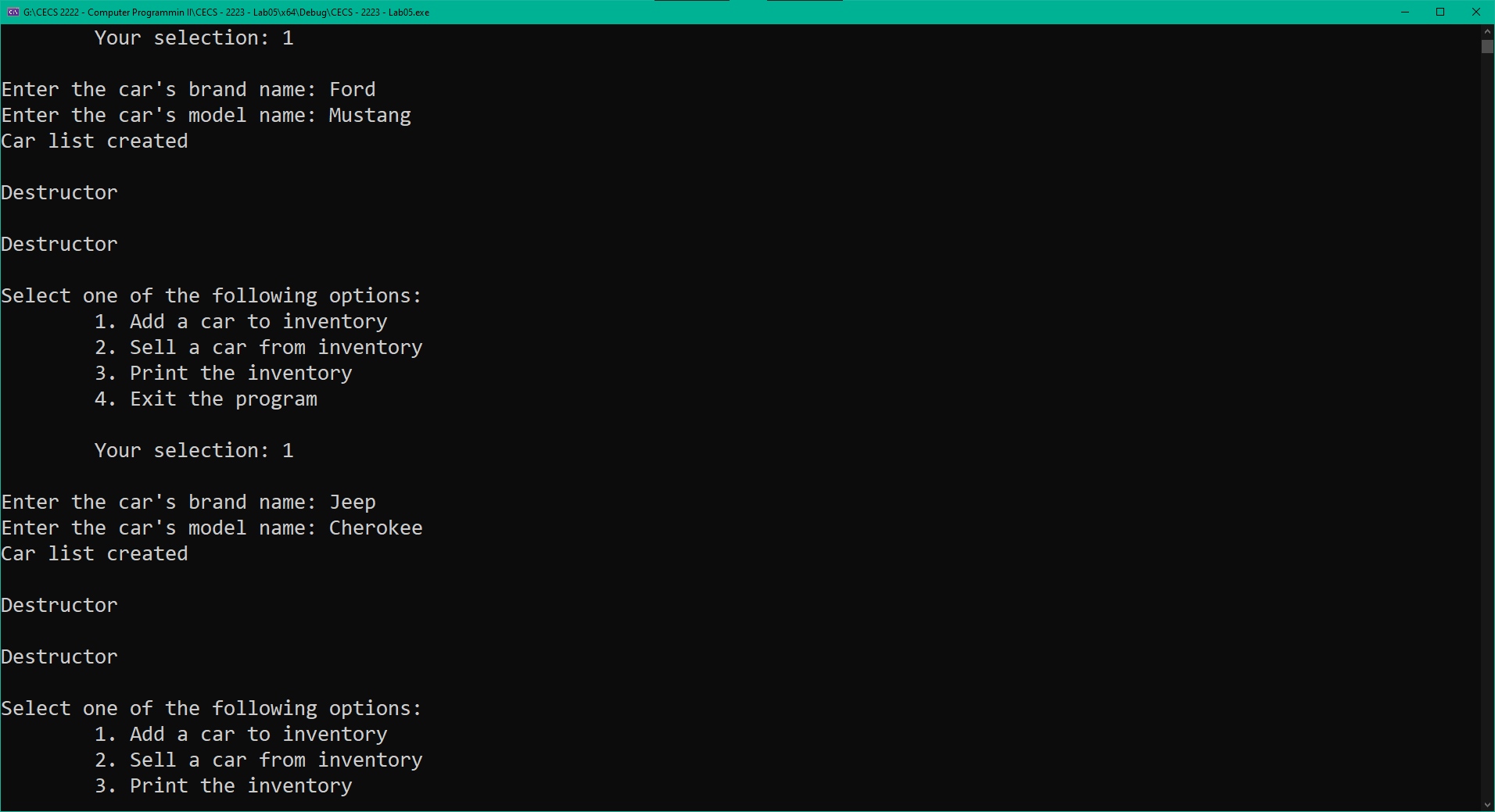
4. Exit the program

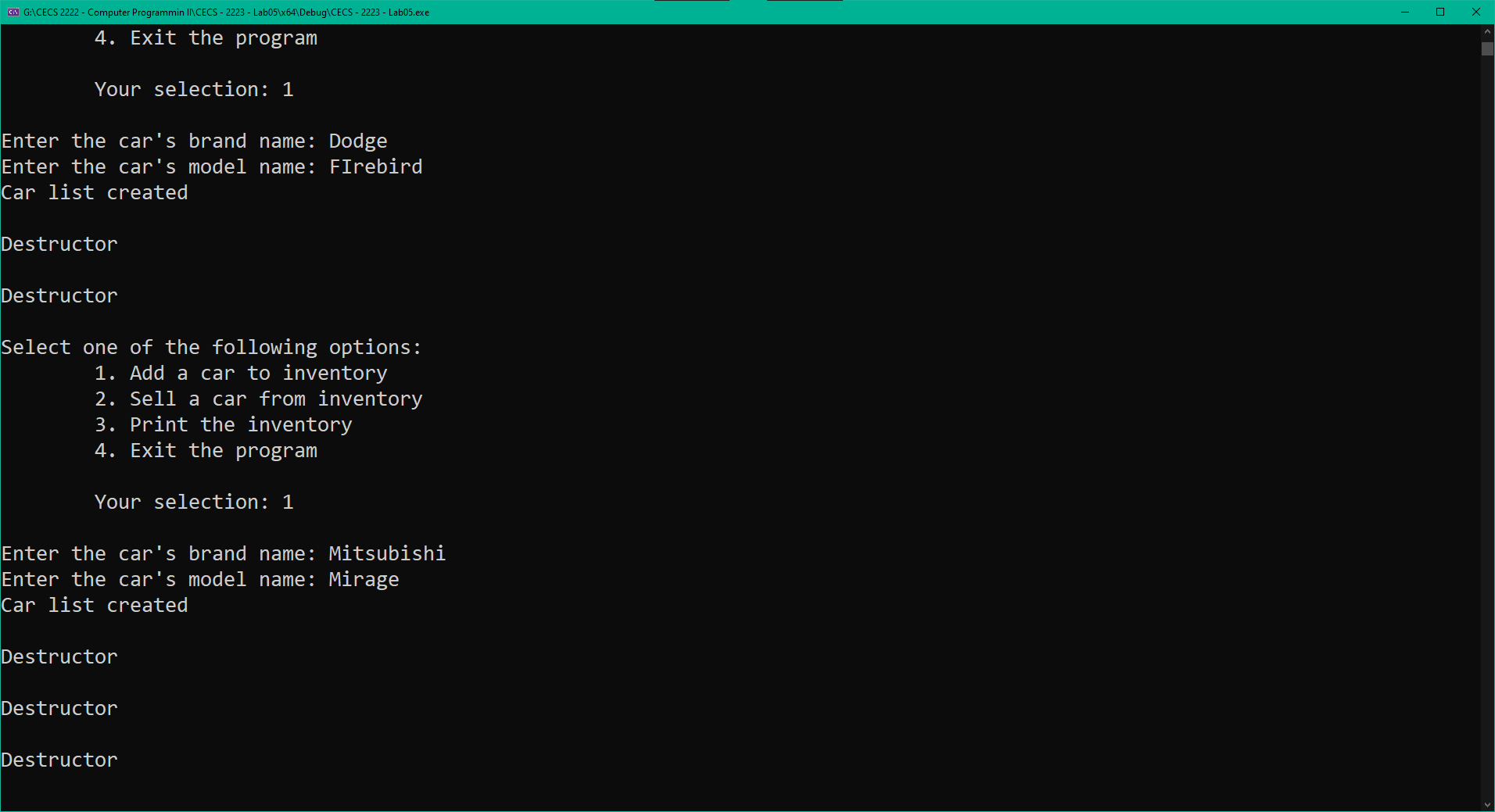
Your selection: 4

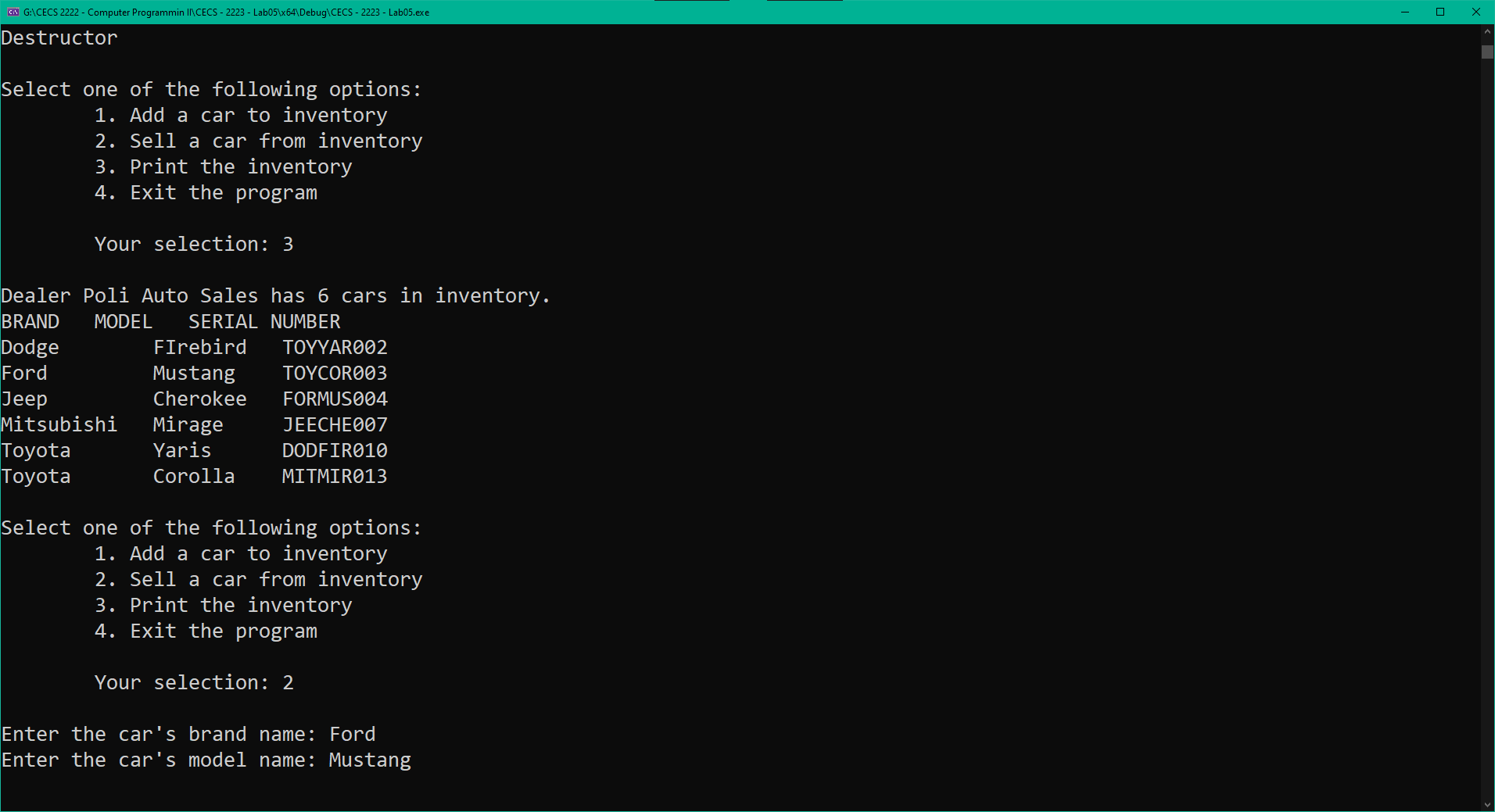
Thank you for using this program!

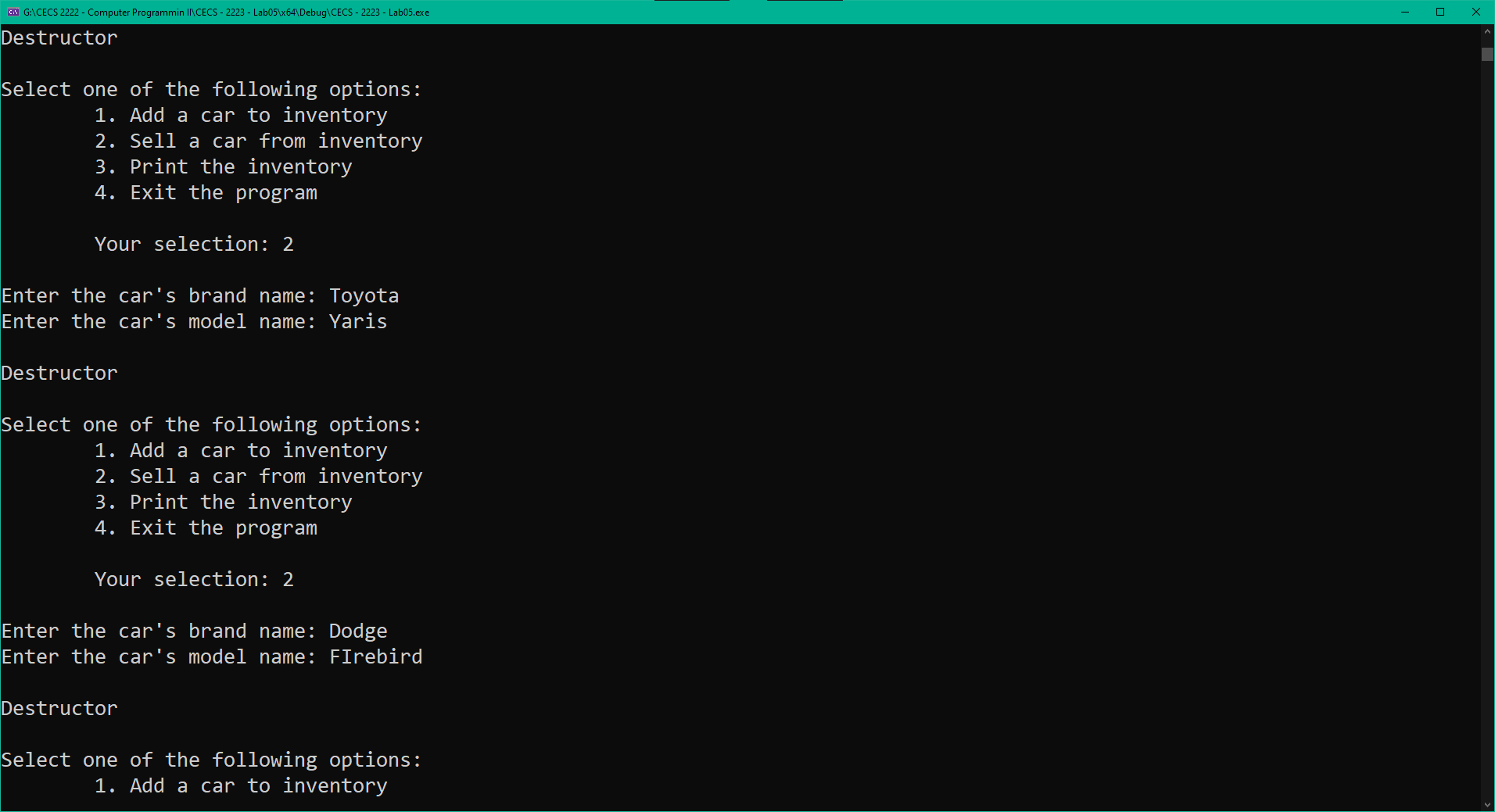
Program developed by Jorge A. Serrano, ID#121260Press any key to continue . . .

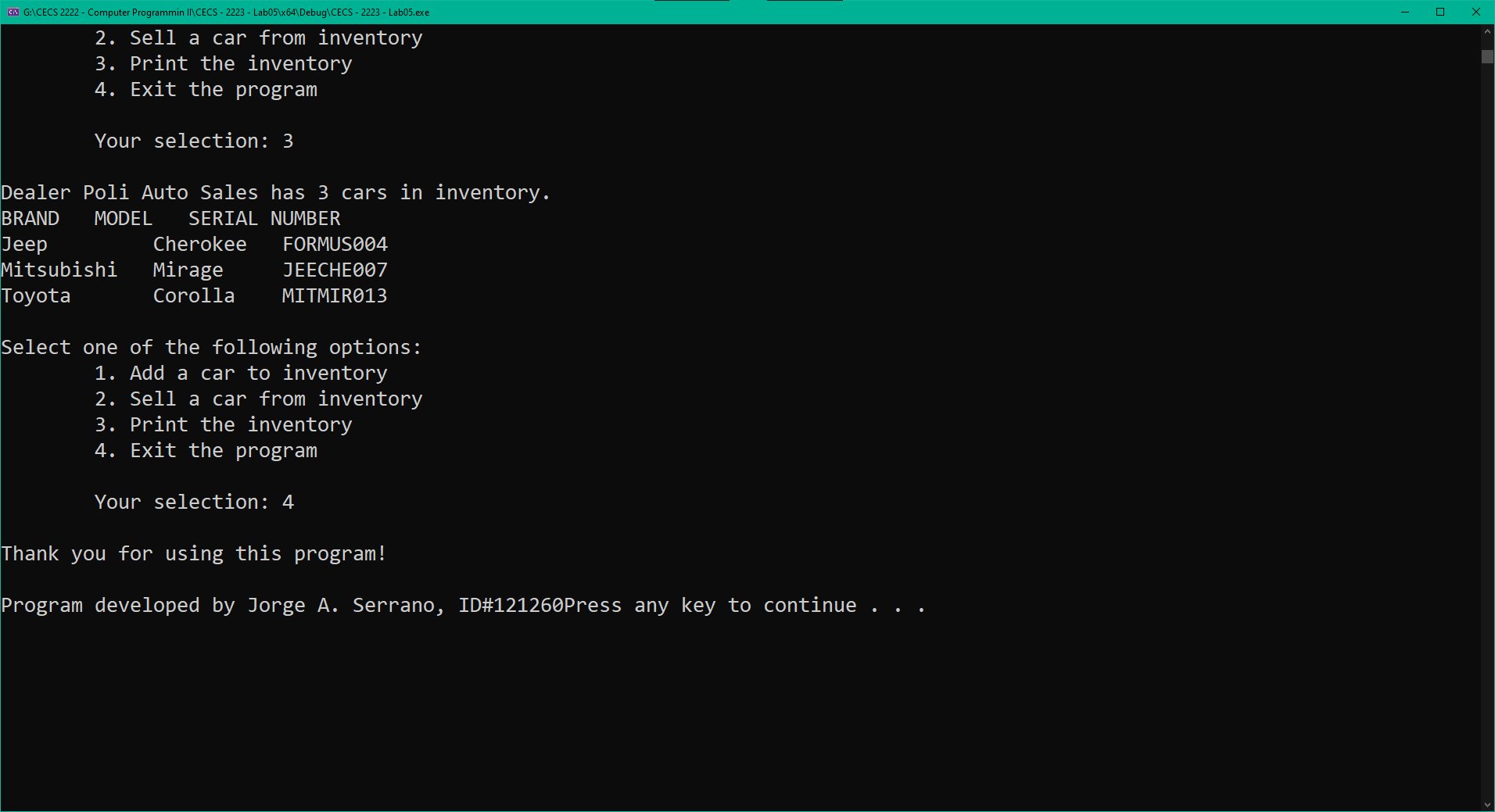


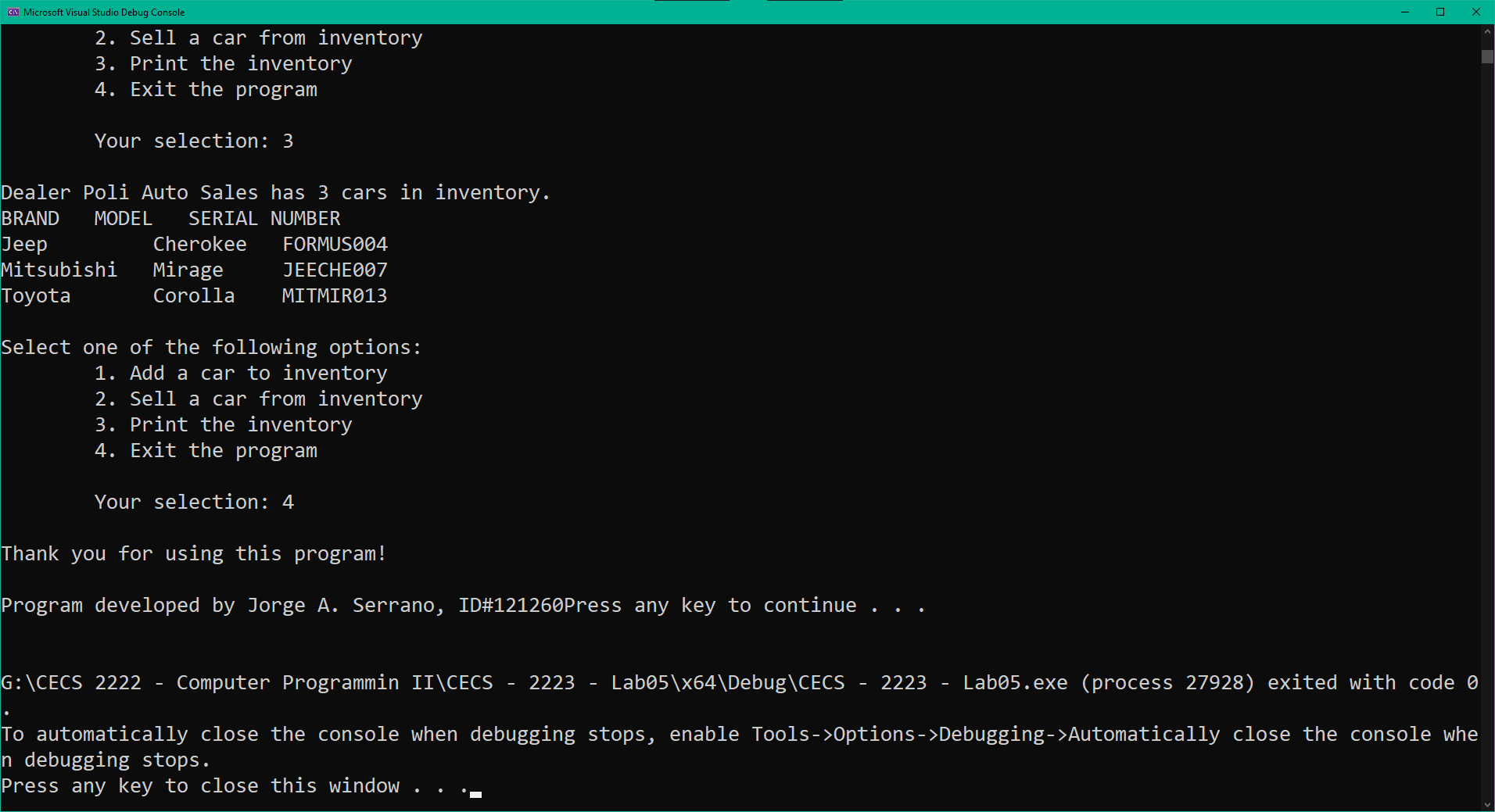












1. Comment on any warnings or errors revealed by Visual Studio. If any error messages were present, list the error and describe how you corrected it. (*5 points*)

I had an error message but it was a link error message. I managed to fixed it in the void Dealer::sellCar(string carBrand, string carModel) method. In the if (carIndex != -1) I had if (carIndex != 0). Apart from that, the serial number doesn’t change with orderInventory, the reason being that the string setSerialNumber(); method is private in the Cars.h file, otherwise the orderInventory would affect the serial number as well.