1. Code

Vehicle.cpp

```
#include "Vehicle.h"
using namespace std;

Vehicle::Vehicle()
{
	age = 0;
	price = 0.0;
}

void Vehicle::setAge(int thisAge)
{
	age = thisAge;
}

void Vehicle::setPrice(float thisPrice)
{
	price = thisPrice;
}

int Vehicle::getAge()
{
	return age;
}

float Vehicle::getPrice()
{
	return price;
}
```

Vehicle.h

```
#pragma once
#ifndef VECHICLE_H
#define VEHICLE_H

#include <iostream>
#include <string>
using namespace std;

class Vehicle
{
public:
	Vehicle();
	void setAge(int thisAge);
	void setPrice(float thisPrice);
	int getAge();
	float getPrice();
private:
	int age;
	float price;
```

```
};

#endif
```

Stub_main_Vehicle.cpp

```
#include "Vehicle.h"
using namespace std;



int main()
{
	Vehicle myVehicle;
	int tempAge;
	float tempPrice;

	do {
		cout << "Vehicle's age: ";
		cin >> tempAge;
		if (tempAge < 0)
		{
			cout << endl << "Invalid input. Vehicle cannot be negative years
old" << endl;
		}
	} while (tempAge < 0);
	myVehicle.setAge(tempAge);

	cout << "Age stored is " << myVehicle.getAge() << endl;


	do {
		cout << "How old is your Vehicle now?: ";
		cin >> tempAge;
		if (tempAge < myVehicle.getAge())
		{
			cout << endl << "Invalid input. Age cannot be less than what is
stored before." << endl;
		}
		else if (tempAge < 0)
		{
			cout << endl << "Invalid input. Vehicle cannot be negative years
old" << endl;
		}
	} while (tempAge < myVehicle.getAge() || tempAge < 0);
	myVehicle.setAge(tempAge);
	cout << "Age of Vehicle stored" << endl;




	do {
		cout << "Vehicle's price: ";
		cin >> tempPrice;
		if (tempPrice < 0)
		{
```

```
                        cout << endl << "Invalid input. Price value cannot be negative" <<
endl;
                }
        } while (tempPrice < 0);
        myVehicle.setPrice(tempPrice);

        cout << "Price stored is $" << myVehicle.getPrice() << endl;



        do {
                cout << endl << "How much is your Vehicle worth now?: ";
                cin >> tempPrice;
                if (tempPrice > myVehicle.getPrice())
                {
                        cout << endl << "Invalid input. You cannot sell your Vehicle more
than its\nprevious worth" << endl;
                }
                else if (tempPrice < 0)
                {
                        cout << endl << "Invalid input. Price value cannot be negative" <<
endl;
                }
        } while (tempPrice > myVehicle.getPrice() || tempPrice < 0);
        myVehicle.setPrice(tempPrice);
        cout << "Price of Vehicle stored" << endl;

        return 0;
}
```

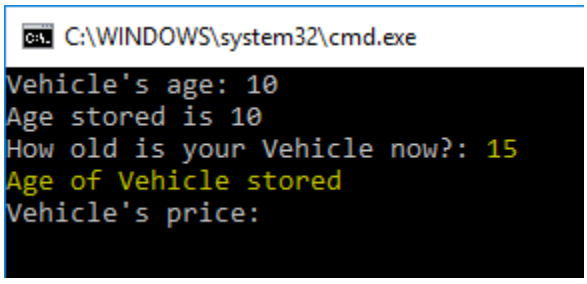
2. Test Plan

Test Strategy	Test Number	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid	1	Age of vehicle is greater than previously stored value	Previously stored variable = 10 New = 15	"Age of vehicle stored"	"Age of vehicle stored"	Pass
Valid	2	Price of vehicle is less than previously stored value unless storing it 1st time	Previously stored variable = 50000 New = 10000	"Price of vehicle stored"	"Price of vehicle stored"	Pass
Valid	3	Price value is always positive	Price = 50,000	"Price stored is $50000"	"Price stored is $50000"	Pass

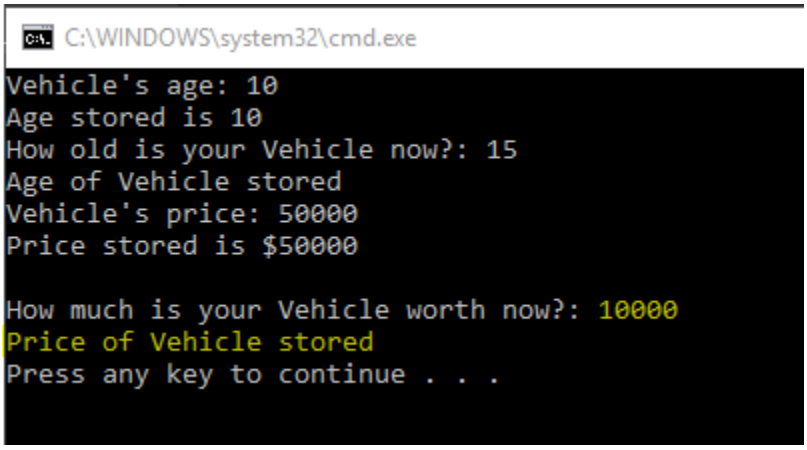
Valid	4	Age value is always positive	Age = 10	"Age stored is 10"	"Age stored is 10"	Pass
Invalid	1	Age of vehicle is more than previously stored value	Previously stored variable = 15 New = 5	"Invalid input. Age cannot be less than what is stored before"	"Invalid input. Age cannot be less than what is stored before"	Pass
Invalid	2	Price of vehicle is more than previously stored value unless storing it 1st time	Previously stored variable = "20,000" New = "30,000"	"Invalid input. You cannot sell your vehicle more than its previous worth"	"Invalid input. You cannot sell your vehicle more than its previous worth"	Pass
Invalid	3	Price value is negative	Price value = -50, 000	"Invalid input. Price value cannot be negative"		

3. Screenshots

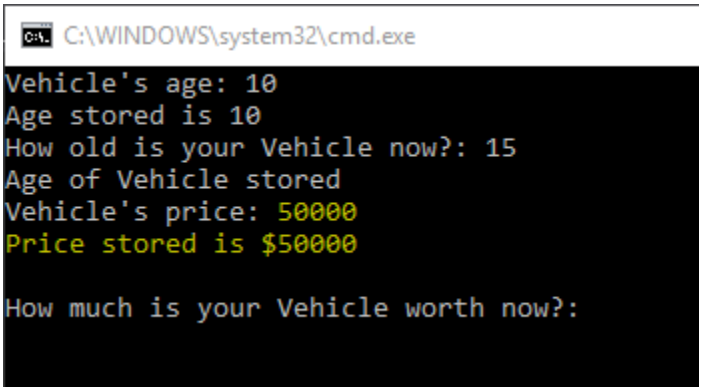
Valid Test Case 1:



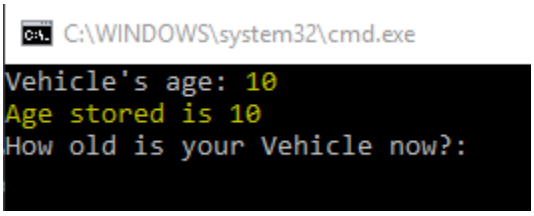
Valid Test Case 2:



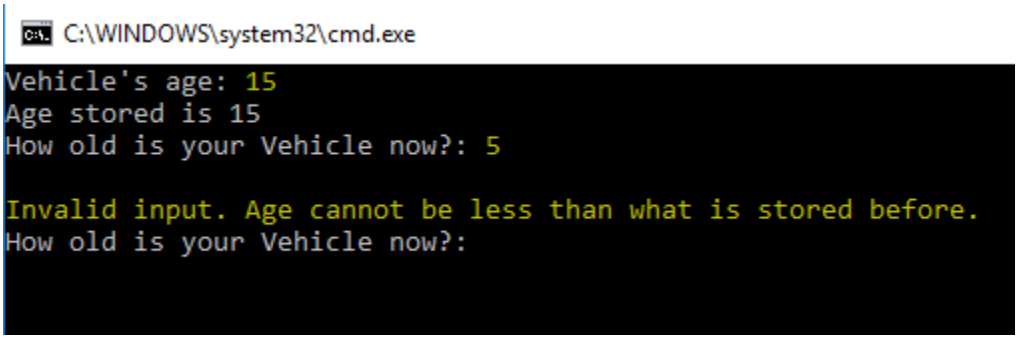
Valid Test Case 3:



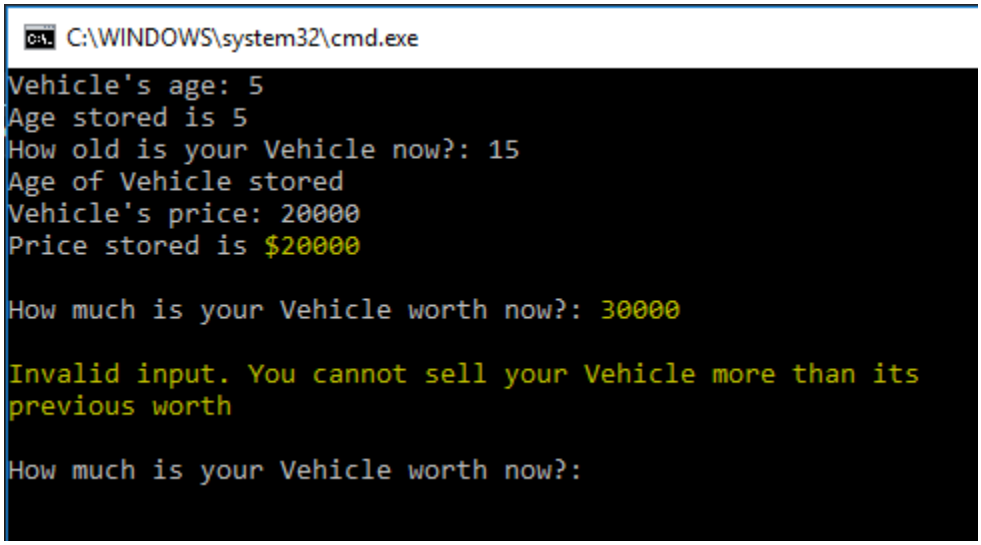
Valid Test Case 4:



Invalid test Case 1:



Invalid Test Case 2:



Invalid Test Case 3: