**NAME: Surya**

**EMP ID: T1934**

**Task Requirements:**

**1.Define a Base Class (AutonomousVehicle):**

The AutonomousVehicle class is the parent class that contains common attributes (like vehicleType, speed, batteryLevel) and abstract methods (navigateRoute and performSelfCheck).#inc

The **navigateRoute** and **performSelfCheck** methods are pure virtual functions, meaning derived classes must implement them**.**

**2.Create Derived Classes:**

ElectricCar: A class that inherits from **AutonomousVehicle**. It provides its own behavior for the **navigateRoute** and **performSelfCheck** methods.

HybridCar: Another class that inherits from **AutonomousVehicle**, with its own implementations of **navigateRoute** and **performSelfCheck**.

AutonomousTruck: A third class that inherits from **AutonomousVehicle** and implements its own **navigateRoute** and **performSelfCheck** behaviors.

**3.Create Objects of Derived Classes:**

In the main() function, create instances of ElectricCar, HybridCar, and AutonomousTruck with specific speed and battery level.

**4.Call Overridden Methods Using Polymorphism:**

Call the **performSelfCheck** and **navigateRoute** methods on the objects. Even though the methods are called on the parent type (AutonomousVehicle), the correct method for each derived class is executed because of polymorphism.

**5.Destructor for Cleanup:**

Each class has a destructor that prints when the object is destroyed, showing the proper cleanup of each object.

Ans)

#include <iostream>

using namespace std;

class AutonomousVehicle

{

protected:

string vehicleType;

int speed;

int batteryLevel;

public:

AutonomousVehicle(string type, int spd, int battery)

{

vehicleType = type;

speed = spd;

batteryLevel = battery;

}

public:

virtual void navigateRoute() = 0;

virtual void performSelfCheck() = 0;

~AutonomousVehicle()

{

cout << "Base class Distructor is called " << endl;

}

};

class ElectricCar : public AutonomousVehicle {

public:

ElectricCar(int spd, int battery) : AutonomousVehicle("Electric Car", spd, battery) {}

void navigateRoute() override {

cout << vehicleType << " Eletric car is navigateing the route" << endl;

}

void performSelfCheck() override {

cout << vehicleType << "Electric vehicle self checking battery level" << batteryLevel << endl;

}

};

class HybridCar : public AutonomousVehicle {

public:

HybridCar(int spd, int battery) : AutonomousVehicle("Hybrid Car", spd, battery) {}

void navigateRoute() override {

cout << vehicleType << " Hybrid car is navigating the route: " << endl;

}

void performSelfCheck() override {

cout << vehicleType << " Hybrid car perform self checking battery level: " << batteryLevel << endl;

}

};

class AutonomousTruck : public AutonomousVehicle {

public:

AutonomousTruck(int spd, int battery) : AutonomousVehicle("AutonomousTruck", spd, battery) {}

void navigateRoute() override {

cout << vehicleType << " AutonomousTruck is navigating the route : " << endl;

}

void performSelfCheck() override {

cout << vehicleType << " AutonomousVehicle can perform self checking battery level " << batteryLevel << endl;

}

};

int main()

{

AutonomousVehicle\* vehicles[] = {

new ElectricCar(80, 90),

new HybridCar(60, 75),

new AutonomousTruck(50, 85)

};

for (int i = 0; i < 3; i++) {

vehicles[i]->navigateRoute();

vehicles[i]->performSelfCheck();

delete vehicles[i];

}

return 0;

}