class Car

{

private:

static int totalCars;

char\* name;

bool isElectric;

int topSpeed;

public:

Car(const char\* n = "honda", bool e = false, int s = 200)

{

int length = strlen(n);

name = new char[length + 1];

int i = 0;

for (; i < length; i++)

name[i] = n[i];

name[i] = '\0';

isElectric = e;

topSpeed = s;

totalCars++;

}

~Car()

{

delete[] name;

}

Car(const Car& rhs) // Copy Constructor with shallow copy

{

name = rhs.name;

isElectric = rhs.isElectric;

topSpeed = rhs.topSpeed;

totalCars++;

}

Car& operator=(const Car& rhs) // Assignment Operator with deep copy

{

if (&rhs != this)

{

int length = strlen(rhs.name);

name = new char[length + 1];

int i = 0;

for (; i < length; i++)

name[i] = rhs.name[i];

name[i] = '\0';

isElectric = rhs.isElectric;

topSpeed = rhs.topSpeed;

}

return \*this;

}

Car operator+(const Car& rhs)

{

Car temp = rhs;

temp.topSpeed += topSpeed;

return temp;

}

bool operator!()

{

return !isElectric;

}

void upgradeCar(const Car& rhs)

{

if (rhs.topSpeed <= topSpeed)

cout << "Error. Cannot upgrade!";

else

topSpeed = rhs.topSpeed;

}

void printInfo()

{

cout << "Car name is " << name << "\n";

if (isElectric)

cout << "It is an electric car. \n";

else

cout << "It is not electric powered. \n";

cout << "Its top speed is " << topSpeed << "\n";

cout << endl << endl;

}

};

int Car::totalCars = 0;