1 Inheritance, Lab Exercise 4 - Cars

Develop a class **Racecar** that inherits **public**ly from class **Car**, which represents a car by its maximum speed, the number of engine valves, its color and its name. A **Racecar** is distinguished by its gearbox (the number of gears it has), its sponsor and the presence of a parachute. Provide only *set* methods for the **private** data. The only way to view the **private** data should be through the **print** method.

The output should appear as follows:

```
chevy:
Car: Chevrolette is black and has a 4-valve engine. MAX SPEED = 95 mph.

f1:
Car: Ferrari is red and has a 40-valve engine. MAX SPEED = 360 mph.
Ferrari also has 7 gears and is sponsored by Scuderia.
Ferrari has used its parachute.
```

```
//car.h
#ifndef CAR_H
#define CAR_H
#include <iostream>
using std::ostream
class Car {
public:
    Car( const char *name, const char *color);
    ~Car();
    void setMaxSpeed( int );
    void setEngineValve( int );
    void print() const;
protected:
    int maxSpeed;
    int engineValves;
    char *colors;
    char *name;
};
#endif
-----------------------------------------------------
//car.cpp
#include <iostream>
using std::cout;
using std::endl;
#include <cassert>
#include "car.h"
/* write the constructor for Car, which takes the Car's name and
```

```
color and assigns them to private data members name and color;
initialize maxSpeed to 95 and engineValves to 4*/
Car::~Car()
{
    delete [] name;
    delete [] color;
}
void Car::setMaxSpeed( int s )
{
    maxSpeed = ( ( s >= 0 && s < 250 ) ? s : 40 );
}
void Car::setEngineValves( int v )
{
    engineValves = ( ( v >= 0 && v < 50 ) ? v : 4 );
}
void Car::print() const
{
    cout << "Car: " << name << " is " << color << " and has a "
         << engineValves << "-valve engine. MAX SPEED = "
         << maxSpeed << " mph. " << endl;
}
-----------------------------------------------------
// racecar.h
#ifndef RACECAR_H
#define RACECAR_H
#include "car.h"
/* write class header for Racecar, which inherits publicly from
Car */
public:
    Racecar( const char *, const char *, const char *);
    ~Racecar();
    void setGearbox( int );
    void useParachute( void );
    void print() const;
private:
    int gearbox; // the number of gears in a car (i.e., 5-speed)
    char *sponsor;
    bool parachuteDeployed;
};
#endif
-----------------------------------------------------
// racecar.cpp
#include <iostream>
```

```
using std::cout;
using std::endl;
#include <cassert>
#include "racecar.h"
Racecar::Racecar( const char *n, const char *c, const char *s )
    /* write code to call base-class constructor */
{
    /* write code to allocate memory for the sponsor, s, and
    copy it into private data member sponsor */
    gearbox = 6;
    parachuteDeployed = false;
}
Racecar::~Racecar()
{
    delete [] sponsor;
}
void Racecar::setGearbox( int gears )
{
    gearbox = ( ( gears <= 10 && gears >= 0 ) ? gears : 6 );
}
void Racecar::useParachute( void ) { parachuteDeployed = p;}
void Racecar::print() const
{
    /* call base class method print here */
    cout << name << " also has " << gearbox
         << " gears and is sponsored by " << sponsor << ". ";
    if ( parachuteDeployed )
        cout << /* access base class version of name here */
             << " has used its parachute." << endl;
    else
        cout << /* access base class version of name here */
             << " has not used its parachute." << endl;
}
-----------------------------------------------------
// driver for race car and car
#include <iostream>
using std::cout;
using std::endl;
#include "car.h"
#include "racecar.h"
int main()
{
    Car chevy( "Chevrolette", "black");
    cout << "chevy: \n";
    /* write code to print Car object */
    Racecar f1( "Ferrari", "red", "Scuderia" );
    f1.setEngineValves( 40 );
    f1.setMaxSpeed( 360 );
    f1.setGearBox( 7 );
    f1.useParachute();
    cout  << "\n\nf1: \n";
    f1.print();
    return 0;
}
-----------------------------------------------------
```

Tips:

- Be sure to allocate sufficient memory when copying character arrays.
- The **Racecar** constructor should explicitly call the constructor of **Car**.

Questions

1. What happens if **Car**'s data members are specified as **private** instead of **protected**?
2. Could **Racecar** have been derived from **Car** using **protected** inheritance? What about **private** inheritance?
3. Is it possible to call class **Car**'s **print** method if you have a **Racecar** object? How does the compiler determine which **print** method to use?
4. Why is it not necessary for **Racecar**'s **print** method to use **Car::name** in order to print the name of the racecar?