CIS 22B Lab 3

Topics:

Introduction to classes (continued)

Overloading function names

Abstract data types

Friend functions

Overloading operators

Use the same format for problem and function headings as assignment 1.

Problem 3.1

Copy the solution from problem 2.2

Make the following additions and changes:

1. Arrange the functions in the order:

\* main

\* output

\* setUp (was setUpCar)

\* input

2. Change the output function so it is a **member function** of the class Car.

3. Change the name of the setUpCar function to setUp and make it a

**member function** of the class Car, with five parameters specified as

constant reference parameters.

4. Make all the data in the Car class private and make the member

functions public.

5. Do **NOT** make the input function a member function.

6. Change the parameters of the input function to five reference

parameters corresponding to the five fields in the class.

7. In main define five variables. Pass these variables to the input

function so the input function can set their values.

8. In main define a Car object. This time the Car object will be in the

stack for main, rather than in the heap. Name this object car1.

9. In main call the setUp function and pass the five variables, rather

than calling it from the input function.

10. In main call the output function.

11. Test the program with the same data as was used in assignment 2.

Problem 3.2

Copy the solution from problem 3.1

Make the following additions and changes:

1. Build three constructors and a destructor for your Car class:

\* A constructor that accepts five constant reference parameters.

Build this constructor with only one line of code that calls the

setUp member function.

\* A copy constructor.

\* A default constructor that sets the following values:

reportingMark (an empty string)

carNumber 0

kind other

loaded false

destination NONE

\* A destructor that does nothing, basically a stub.

2. Arrange the constructors and destructor in the class before the

other member functions in the order:

default constructor, copy constructor, constructor that takes five

constant reference parameters, destructor

3. Arrange the function definitions in the same order.

4. Remove the call to the setUp function from the main function.

5. Revise the main function to create three Car objects in the stack:

6. car1 with the same values used in assignment B.

7. car2 which is a copy of car1.

8. car3 which is created with the default constructor.

Now, add code in the main function to print these headings and use the

Car member function to print the data for each of the three cars:

Contents of car1:

reportingMark: SP

carNumber: 34567

kind: box

loaded: true

destination: Salt Lake City

Contents of car2:

reportingMark: SP

carNumber: 34567

kind: box

loaded: true

destination: Salt Lake City

Contents of car3:

reportingMark:

carNumber: 0

kind: other

loaded: false

destination: NONE

Problem 3.3

Copy the solution from problem 3.2

Create a friend function for the function operator== which tests to see

if two objects are equivalent. The two objects are equivalent if they

have the same reportingMark and number (do not look at the kind, loaded,

and destination fields).

Test with the following code in main:

if (car1 == car2)

cout << "car1 is the same car as car2\n";

else

cout << "car1 is not the same car as car2\n";

if (car2 == car3)

cout << "car2 is the same car as car3\n";

else

cout << "car2 is not the same car as car3\n";