CS 31 Worksheet 5

This worksheet is entirely **optional**, and meant for extra practice. Some problems will be more challenging than others and are designed to have you apply your knowledge beyond the examples presented in lecture, discussion or projects. All exams will be done on paper, so it is in your best interest to practice these problems by hand and not rely on a compiler.
Solutions are written in red. The solutions for **programming** problems are not absolute, it is okay if your code looks different; this is just one way to solve the specific problem.

Concepts: Structs/Class, Public/Private, Simple constructors, Dynamically allocating individual objects

1. Conceptual Questions
    - What's the main difference between declaring a type with the keyword `struct` and declaring it with the keyword `class`?

      `struct` has everything public by default if you don't specify otherwise, whereas class has everything private by default.

    - Why should you not allow data members to be public?

      You do not want other users to be able to directly manipulate your object's data members (e.g., because they could set them to invalid values). By making data members private and providing a public interface that they must use, your implementation of that interface can control what values data members are set to.

    - What is the purpose of having private member functions in a class? Can you give some examples of when they would be used?

      Private functions are useful for placing code that the user does not need to be aware of but will be helpful for the implementations of your member functions. One example is a helper function with code that is common to two or more member functions.

    - (True/False) A class may have more than one constructor.

      True. You can overload constructors (as long as they differ in the number or type of arguments).

2. Write a class Person that has two private data members:
    - `m_age` (an int)

- `m_catchphrase` (a string).

The Person class should have a default constructor that initializes its member variables to reasonable values and a second constructor that initializes the member variables to the values of its parameters. In addition, Person should have three public member functions:

- `getAge()`, which returns the Person's age
- `haveBirthday()`, which increments the Person's age by 1
- `speak()`, which prints the Person's catchphrase.

```
class Person {
 public:
  Person() {
    m_age = 0;
    m_catchphrase = "";
  }
  Person(int age, int catchphrase) {
    m_age = age;
    m_catchphrase = catchphrase;
  }
  int getAge() const {
    return m_age;
  }
  void haveBirthday() {
    m_age++;
  }
  void speak() const {
    cout << m_catchphrase << endl;
  }
 private:
  int m_age;
  string m_catchphrase;
};
```

3. A line in Euclidean space can be represented by two parameters, **m** and **b** from its slope-intercept equation **y = mx + b.** Here **m** represents the slope of the line and **b** represents the line's y-intercept.
   Write a class that represents a line. Your class must have a simple constructor that initializes the line's **m** and **b**. Next, define a member function with the following prototype:
   `double intersection(Line line2);`

   This function must compute the x-coordinate where this line and another line (line2) intersect.

```
double m1 = 2;
double b1 = 3;
double m2 = -2;
```

```
double b2 = 7;
Line line1(m1, b1);
Line line2(m2, b2);
cout << line1.intersection(line2) << endl;    // prints 1.0
```

This function must compute the x-coordinate where this line and another line (line2) intersect.

```
class Line {
public:
  Line(double m, double b) {
    m_m = m;
    m_b = b;
  }

  double m() const {
    return m_m;
  }

  double b() const {
    return m_b;
  }
  double intersection(Line line2) {
    if (m_m == line2.m()) {
      // same slope! SO the lines either are coincident or parallel
      // spec doesn't specify what we should do here, so return
      // whatever; in the real world we may want to throw an exception
     // (which aren't discussed in CS 31)
       return 0;
    }
    return (line2.b() - m_b)/(m_m - line2.m());
  }
private:
  double m_m;
  double m_b;
};
```

Bonus: There are two or three ways in which this problem specification is incomplete; they are not related to C++, but to the problem domain. What are they?

As mentioned in the comments above, the spec does not tell us what we should return if the two lines are coincident or parallel. Also, vertical lines cannot be exactly defined using the framework we have (e.g. x=3), although they can be approximated using a line with a large m.

4. Find the **three** errors in the following code, and write the fixes.

```
class Cat {
      int m_age;
      string m_name;
      string m_type;
   public: (1)
      Cat(int age, const char name[], string type) {
            m_age = age;
            m_name = name;
            type = type;
      }
      void introduce() {
            cout << "Hi! I am a " + type + " cat" << endl;
      }
};

struct Sheep {
      string m_name;
      int m_age;
      Sheep(int age) {
            m_age = age;
      }
      void introduce() {
            cout << "Hi! I am " + m_name + " the sheep" << endl;
      }
};(2)
Don't forget the semicolon! (2)

int main() {
      Cat schrodinger(5, "Schrodinger's cat", "Korat");
      Schrodinger.introduce();
      cout << schrodinger.m_age << endl; (3)
      Private variables cannot be accessed outside a class
      declaration. (3)

      Sheep dolly(6);
      dolly.introduce();

}
```

The errors are highlighted in **red** and the fixes are in **green**! Each error and fix can be matched by the corresponding number.

What will the program above successfully print once all the fixes have been made?

Hi! I am a Korat cat!
Hi! I am the sheep!

We never initialized the name data member in Sheep, and the default constructor of a string makes it an empty string; this explains our result.

5. Write a class called Complex, which represents a complex number. Complex should have a default constructor and the following constructor:

```
Complex(int real, int imaginary);
// -3 + 8i would be represented as Complex(-3, 8)
```

Additionally, the class should contain two functions: sum and print. Sum should add two complex numbers. Print should print which complex number the object represents. You may declare any private or public member variables or getters/setters you deem necessary. Your code should work with the example below.

```
int main() {
(1)   Complex c1(5, 6);
(2)   Complex c2(-2, 4);
(3)   Complex c3;

(4)   c1.print();
(5)   c2.print();
(6)   cout << "The sum of the two complex numbers is:" << endl;
(7)   c3.sum(c1, c2);
(8)   c3.print();
}

// The output of the main program:
5+6i
-2+4i
The sum of the two complex numbers is:
3+10i

class Complex {
    int m_real;
    int m_imaginary;
public:
    Complex() {}
    Complex(int real, int imaginary) {
        m_real = real;
        m_imaginary = imaginary;
```

```
    }
    void print() {
        cout << m_real << "+" << m_imaginary << "i" << endl;
    }
    void sum(Complex c1, Complex c2) {
        m_real = c1.m_real + c2.m_real;
        m_imaginary = c1.m_imaginary + c2.m_imaginary;
    }
};
```