

CS 31 Worksheet Week 8

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.

If you have any questions or concerns please contact your LA or go to any of the LA office hours.

Concepts: Classes and Structs

1. Conceptual Questions

- What's the main difference between declaring a type with the keyword `struct` and declaring it with the keyword `class`?
- Why should you not allow data members to be public?
- What is the purpose of having private member functions in a class? Can you give some examples of when they would be used?
- What happens if you forget to deallocate memory once you're done with the object?
- (True/False) A class may have more than one constructor.
- (True/False) A class may have more than one destructor.
- If you have an object pointed to by a pointer, which operator is used with the pointer to access the object's members?

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 data members to reasonable values and a second constructor that initializes the data members 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.

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.

4. Write a program that **repeatedly** reads an age and a catchphrase from the user and uses them to **dynamically** allocate a `Person` object, before calling the `Person`'s `speak()` function and then **deallocating** the `Person` object.

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`. Calling `sum` should set the calling object to the sum of the 2 complex numbers

passed as arguments. 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 = new Complex();  
  
  (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();  
  (9)  delete c3;  
}
```

// The output of the main program:

5+6i

-2+4i

The sum of the two complex numbers is:

3+10i

Bonus: What would happen if we swapped line (8) and (9)?