Lauren Lyne

This homework includes short answer questions and a programming assignment. The short answer questions are worth 5 points each. The programming assignment is worth 30 points. The short answer questions may cover topics that are in the text, but were not covered in the lectures. Push a preliminary version of the programming assignment to Github before Wednesday's class. Push the final version of your source code, Makefile, and a text file with your short answer questions to Github before 5PM Friday.

1. What is wrong with the following code and how would you fix it?

```
#ifndef PROJECTILE_H
 #define PROJECTILE_H
 class Projectile {

private:
    double position;
    double velocity;

public:
    Projectile(double position, double velocity);
    virtual ~Projectile();

   double getPosition() const;
    double getVelocity() const;

} // end of Projecile class
 #endif
```

There is no semicolon following the } at the end of the class! In order to fix this you would just add the semicolon to make it };

2. The following is the definition of the constructor for the Projectile class above, but there are three things wrong with it. What are they and how would you fix them?

```
Projectile(int position, int velocity) {
    this.position = position;
    this.velocity = velocity;
```

} // end of constructor

This.”anything” is not a proper thing to use in C, even though it works in java. You can still use “this”, but instead you make it into a pointer. Also, since the constructor would be placed in the main.cc file, you would indicate the class that the method belongs to. It should be written as follows:

```
Projectile::Projectile(int position, int velocity) {
        this->position = position;
        this->velocity = velocity;
}
```

3. Describe each of the following methods

(a) int* method(int* arg);

A method that returns a variable that when dereferenced is an int. The method takes in a variable arg that when dereferenced is an int.

(b) const int* method(int* arg);

A method that returns a variable that when dereferenced is a constant int that can not be changed. The method takes in a variable arg that when dereferenced is an int.

(c) const int* const method(int* arg);

A method that returns a constant variable that when dereferenced is a constant int. The method takes in a variable arg that when dereferenced is an int. The const before the method indicates that the return type must be const.

(d) const int* const method(const int* arg);

A method that returns a constant variable that when dereferenced is a constant int. The method takes in a variable arg that when dereferenced is a constant int. The const before the method indicates that the return type must be const.

(e) const int* const method(const int* arg) const;

A method that returns a constant variable that when dereferenced is a constant int. The method takes in a variable arg that when dereferenced is a constant int. The const before the method indicates that the return type must be const. The const at the end of the line indicates that the method does not modify the object.

4. In what ways are C++ strings better than C strings? In what ways are C strings better than C++ strings?

C++ strings are safer and more convenient to use than C strings. Since a C strings are an array of chars, this leads to security issues. C string are better than C++ things for certain actions such as they are needed to create a string literal, some C libraries that you may need to use have C strings, and writing lightweight and efficient strings to create a super-performance critical code.

5. What is the difference between a pointer and a reference?

A pointer allows you to directly access memory by placing a * in front of or after the variable type. A reference is automatically dereferenced, so you do not need to put the * in to access memory. A reference will always point to the same memory and can not be changed, while you can change what a pointer points to.

6. What is a destructor for?

A destructor is used to delete heap memory allocated by the class. It can also close up files opened by the class.