CMPT 414 / MSCS 565
Game Programming I
Quiz #1

Quiz #2, Form: A

Name: _____

Date: _____

Section 1. Pointers and References

Consider the following, incomplete code sample.

```
1   void demo_ptr_ref() {
2       int a = 100;
3       int b = 200;
4       int& tmp = a;
5       a = b;
6       b = tmp;
7       int* x = ________; // TODO allocate a fresh integer on the heap
8       int* y = ________; // TODO point to the integer b
9       int* z = ________; // TODO make this point to what x points to
10      x = y;
11      y = z;
12      ________ = 500;   // TODO replace the value that x points to
13      return; // Assume control reaches this line for your memory diagram
14  }
```

1. **(Weight=1)** Fill in the missing code above to implement the logic described in the `TODO` comments.
2. **(Weight=3)** Suppose that this function is called. Draw a memory diagram showing the contents of the stack and heap memory when control has reached (but not yet executed) Line 13. *Note: Assume that the code has been filled correctly to implement the behavior described in the `TODO` comments.*

3. **(Weight=1)** When the function `demo_ptr_ref` returns, the values allocated on the heap will remain, but we no longer have pointers or references to those values.

   (a) What is this situation called?

   (b) Assuming we don't need that data after this function is done, what code should we add before the return statement to avoid this problem?

Section 2. I/O Streams

4. **(Weight=1)** Suppose that we have declared an integer variable named `val` that holds the number `42`. Write two simple C++ statements showing...

   (a) how to print this variable to standard output using the *stream insertion* operator

   (b) how to read a new value into this variable from standard input using the *stream extraction* operator

5. **(Weight=1)** Explain the difference between `std::cout` and `std::cerr`, and when you might use each one. (Hint: Can you remember which one is "buffered" and what that means?)

6. **(Weight=1)** The C++ standard library provides specific stream classes for reading and writing to files.

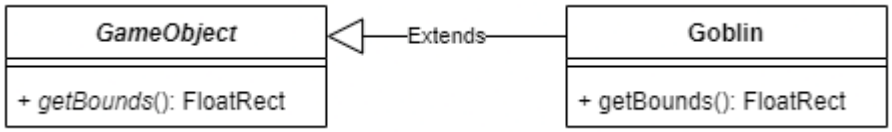
   (a) What C++ header file we must include to use these classes?

   (b) What does the `std::ios_base::openmode` parameter indicate?

Section 3. Foundations of OOP in C++

7. **(Weight=1)** What is the difference between `class` and `struct` in C++?

8. **(Weight=2)** Consider the class diagram below and then answer the questions that follow.



   (a) What keyword must we use in C++ when declaring the `getBounds` method in order to enable *polymorphism* when the derived class `Goblin` overrides this method of its base class `GameObject`?

   (b) Suppose that the base class `GameObject` is abstract and does not implement the `getBounds` method. What do we call such a method in C++?And, how would we write the method declaration in this case?

   (c) Write a few lines of C++ code declaring the `Goblin` class so that it implements the inheritance relationship shown in the diagram. *Note: You do not need to implement the `getBounds` method; assume that will be done separately in the `.cpp` file.*

Section 4. Design Patterns for Games

9. **(Weight=2)** Consider the following pseudo-code from a hypothetical game.

```
while (true) {
    /* Handle input */
    // ...

    /* Update the game state */
    // move the player
    if (player.jumped) {
        player.velocity.y = player.jumpSpeed;
    }
    player.position.x = player.position.x + player.velocity.x * dt;
    player.position.y = player.position.y + (player.velocity.y + gravity) * dt;
    // move the goblin
    goblin.position.x = goblin.position.x + goblin.velocity.x * dt;
    goblin.position.y = goblin.position.y + (goblin.velocity.y + gravity) * dt;
    // move the raindrops
    for (auto& raindrop : weather) {
        raindrop.position.y = raindrop.position.y + gravity * dt;
    }
    // etc...

    /* Render the next frame */
    // ...
}
```

(a) What design pattern can help clean simplify this code so that the main game loop does not have to know all the inner workings of each kind of game object?

(b) Illustrate this design by using either a UML class diagram or C++/pseudo-code.

10. **(Weight=3)** Consider the following code from the *Zombie Arena* game.

```
class TextureHolder {
public:
    TextureHolder() {
        assert(m_s_Instance == nullptr);
        m_s_Instance = this;
    }
    static sf::Texture& GetTexture(std::string const& filename);
private:
    std::map<std::string, sf::Texture> m_Textures;
    static TextureHolder* m_s_Instance;
};
```

(a) This class demonstrates the use of which design pattern?

(b) What problems does this pattern solve for the `TextureHolder` class? In other words, what could happen if we did not use this pattern?

(c) The author uses `assert` to guarantee the uniqueness of `m_s_Instance`. However, this means that we find out only at *run time* if our program attempts to create two `TextureHolder`s. What approach did we study in class that would ensure we would detect this problem at *compile time*?

(d) Give another practical example (besides `TextureHolder`) of a situation where this pattern is useful or even necessary?

(e) Finally, `Singleton` is often overused. What part of the `Singleton` pattern is actually a bad idea in most situations, and why?

11. **(Weight=4)** Demonstrate an understanding of the *Update Method* design pattern.

    (a) First, explain what problem(s) this pattern can help to solve and/or how this pattern is useful in practice.

    (b) Draw a UML class diagram, or write equivalent pseudo-code, to illustrate the *Observer* design pattern. Your diagram (or pseudo-code) should clearly indicate the classes involved and the relationships between them.