Software Engineer C++ Test

Instructions to candidates:

- You are to complete this test within 1 hour.
- This is an open book test (Google probably won't help you much though).
- If any of the questions are unclear, state clearly on what your assumption is and work on the question with those assumptions.
- Non-coding questions should have short and sweet answers. Essays are not allowed.
- If any non-coding answer (with its assumption) is longer than 40 words, it will be ignored and 0 marks will be awarded for that question.
- Coding questions should be answered as pedantically as possible. The code should be optimized as much as you can and it must compile.
- Answers with large amounts of code should be in a .cpp file with the question number as the file name. [E.g. 1.3.cpp will be for question 1.3]
- No marks will be awarded for wrong file names in coding questions.

Submission Guidelines:

- Zip the this document along with all additional files and send it back to us. Ensure that the zip file is renamed as such: <first_name>_<last_name>.zip

Graders Table

C++ Proficiency Test	**Score**
C++ Basics Part I	/4
C++ Basics Part II	/2
Code Analysis	/6
Memory Basics	/2
Multithreading Part I	/3
Multithreading Part II	/3
Advanced C++	/5
Total	/25

Graphics Proficiency Test	**Score**
Graphics Basics Part I	/4
Graphics Basics Part II	/6
Graphics Basics Part III	/5
Advanced Graphics Part I	/5
Total	/20

1.0 - C++ Proficiency Test

1.1 - C++ Basics Part I

What is the difference between the `*has a*` and `*is a*` relation in C++? Give the minimum required code snippet as an example for both.
(4 points)

1.2 - C++ Basics Part II

What is wrong with the following code? Give the minimum required code snippet that corrects this mistake.

```
class Boo{};
class Foo : public Boo{};

int main()
{
	Boo* b1 = new Foo();
	delete b1;
	Return 0;
}
```

(2 points)

1.3 - Code Analysis

Modify the following code snippet so that it compiles and outputs the following:

```
My name is Jack and I own a mouse named Jerry
My name is Jill and I own a cat named Tom
```

DO NOT MODIFY the main function.

```
#include <iostream>
#include <vector>
#include <memory>
#include <string>

class Owner;

class Pet
{
public:
    enum EType { Mouse, Cat };

private:
    Owner &m_owner;
    EType m_type;
    std::string m_name;
};


class Owner
{
public:

private:
    std::string m_name;
    std::shared_ptr<Pet> m_pet;
};


void PrettyPrint(const Owner& owner)
{}
int main()
{
    std::vector<Owner> ownerList;
    ownerList.emplace_back(Owner("Jack", Pet::EType::Mouse, "Jerry"));
    ownerList.emplace_back(Owner("Jill", Pet::EType::Cat, "Tom"));

    for(const auto& owner : ownerList)
    {
        PrettyPrint(owner);
    }
    return 0;
}
```

(6 points)

1.4 - Memory Basics

What is the difference between the stack and heap? When should we prefer one over the other?
(2 points)

1.5 - Multithreading part I

Make the following code snippet thread-safe and output the following:

```
odd: 1
even: 2
odd: 3
even: 4
odd: 5
```

DO NOT MODIFY the main function.
(3 points)

-

```
#include <thread>

#include <iostream>



int maxCount = 5;
int counter = 1;

void even()
{

}

void odd()
{

}

int main()
{
    std::thread thread1(even);
    std::thread thread2(odd);
    thread1.join();
    thread2.join();
}
```

1.6 - Multithreading part II

What is wrong with the following code. What is the name of this problem? Give another scenario where this occurs.

```
#include <thread>

void fun(std::thread* t, bool& locked)
{
	while (locked) {}
	t->join();
}

int main()
{
	std::thread t1;
	std::thread t2;

	bool locked = true;

	t1 = std::thread(fun, &t2, std::ref(locked));
	t2 = std::thread(fun, &t1, std::ref(locked));

	locked = false;

	return 0;
}
```

(3 points)

1.7 - Advanced C++

Describe all errors and optimization problems with the following code snippet. For each of the errors provide a solution.

```
#include <string>
#include <vector>
#include <utility>
#include <iostream>

template<typename T>
std::vector<T>&& makeVector(std::initializer_list<T> init_list)
{
    return std::vector<T>(init_list);
}

int main()
{
    std::vector<std::string>&& sv = makeVector({"foo", "bar"});
     std::cout << sv.size();
    return 0;
}
```

(5 points)

2.0 - Graphics Proficiency Test

2.1 - Graphics Basics Part I

What is the difference between a vertex shader and pixel shader? Explain what each one does.
(2 points)

Given a texture of size 1024x1024 that is loaded into memory, what is the size in MB that is used?
(1 point)

What is the inverse of an Orthogonal matrix?
(1 point)

2.2 - Graphics Basics Part II

State 3 methods to optimize rendering. For each method, briefly explain how it works.
(6 points)

2.3 - Graphics Basics Part III

What is a transform matrix in graphics and what are the 3 major components that make it?
(1 point)

Describe the steps and operations you would need to convert a model's vertices to a pixel position. (You do not need to show the formulas for each operation and can use matrix and pseudo code symbology)
(4 points)

2.4 - Advanced Graphics Part I

What is BRDF? Highlight the 4 major parameters in BRDF and what they represent.
(4 points)

What is meant by Physically Based Rendering and what does it allow us to do?
(1 point)