SMOKING GUN™
INTERACTIVE

Programmer – C++ Evaluation

The sections in this test build upon each other to form one solution for submission. The work has been divided into the three logical steps below.

Programmer Evaluation – Step 1 of 3

Design and implement a C++ program with the following:

- An axis-aligned Rectangle2D with floating-point coordinates.
- An algorithm that checks if two Rectangle2Ds intersect.
- Test code that verifies your implementation's correctness.

Programmer Evaluation – Step 2 of 3

Given a 2D Entity where each has a Rectangle2D bounding box and optional Components:

- Add a Rectangle2D bounding box member to the Entity.
- Add a collection of Components to the Entity. An Entity may have any number of Components. Assume there will be more than the three example Components below.
- Write the implementation to make Entity copyable.
- Feel free to modify the existing code as desired.

```
class Component
{
public:
    virtual ~Component() = default;

    // You may add to the interface.

};

// A few basic Components. Entities can have up to one of each Component type.
// Assume they will have more methods, and there will be more Components: those
// details are not important for this test.
// Add anything needed to support your implementation.

class HealthComponent : public Component
{
private:
    int m_health = 0;
};

class AttackComponent : public Component
{
private:
    unsigned int m_attackPower = 0;
};

class MovementComponent : public Component
{
private:
    float m_speed = 0;
};

class Entity
{
    // TODO: Add:
    //  - A Rectangle2D bounding-box.
    //  - A collection of Components.
    //  - An implementation to make Entity copyable.
};
```

Programmer Evaluation – Step 3 of 3

Using the provided code below as a starting point (though you are not required to use it), create an algorithm that:

- Reads from the included file containing Rectangle2D and Component information to create a collection of Entities.
- Create an algorithm to determine how many Entity intersections are in the collection. Look for unique intersections – if two Entities intersect each other, that counts as one unique intersection rather than two. Use your intersection implementation from Part 1 and the Entity's Rectangle2D bounding box.

```
#include <chrono>
#include <fstream>
#include <iostream>
#include <string>

int main( int argc, const char* argv[] )
{
    if ( argc < 2 )
    {
        std::cerr << "Specify a file to run this program.\n";
        return 1;
    }

    std::cout << "Running program against file: " << argv[1] << "\n";

    std::ifstream file( argv[1], std::ios::binary );
    if ( !file.is_open() )
    {
        std::cerr << "Failed to open file.\n";
        return 2;
    }

    unsigned int numberOfEntities = 0;
    if ( !( file >> numberOfEntities ) )
    {
        std::cerr << "Failed to get number of Entities from file.\n";
        return 3;
    }

    // TODO: Add a collection of Entities.

    for ( unsigned int i = 0; i < numberOfEntities; ++i )
    {
        float x = 0;
        float y = 0;
        float width = 0;
        float height = 0;
```

```
        if ( !( file >> x >> y >> width >> height ) )
        {
            std::cerr << "Error getting bounds on line " << i + 1 << ".\n";
            return 4;
        }

        // Not every Entity has Components.
        std::string componentTypes;
        const auto currentPos = file.tellg();
        file >> componentTypes;
        if ( !componentTypes.empty() && !std::isalpha( componentTypes.back() ) )
        {
            file.seekg( currentPos );
            componentTypes.clear();
        }

        for ( const char type : componentTypes )
        {
            switch ( type )
            {
                case 'H':
                    // TODO: This Entity has a HealthComponent.
                    break;
                case 'A':
                    // TODO: This Entity has an AttackComponent.
                    break;
                case 'M':
                    // TODO: This Entity has a MovementComponent.
                    break;
                default:
                    std::cerr << "Unknown Component type: " << type << "\n";
                    break;
            }
        }

        // TODO: Use the above information to create an Entity with
        // a Rectangle2D bounding box and given Components.

    }
    file.close();

    const auto start = std::chrono::high_resolution_clock::now();

    // TODO: Algorithm to detect number of Entity intersections.

    const auto end = std::chrono::high_resolution_clock::now();
    const auto runMS =
        std::chrono::duration_cast<std::chrono::milliseconds>( end - start );

    std::cout << "Algorithm executed in " << runMS.count() << "ms.\n";
    return 0;
}
```