Object-Oriented Software Development Using C++ OOP345NDD.05494.2214 Practice/Sample Test

Review Test Submission: Midterm Sample #4: Debugging

Review Test Submission: Midterm Sample #4: Debugging

User	Yihsun Lee
Course	Object-Oriented Software Development Using C++
Test	Midterm Sample #4: Debugging
Started	6/21/21 4:32 PM
Submitted	6/21/21 5:13 PM
Status	Needs Grading
Attempt Score	Grade not available.
Time Elapsed	40 minutes out of 40 minutes

Instructions

The **Debugging** questions are presented as C++ code that has errors in it: errors caught by a compiler, generate memory leaks or crashes/undefined behaviour at runtime. Your task is to identify each error by file name and line number and **to explain why the error appears, what C++ standard rule is broken or C++ feature is misused and how to fix the error**. Your answer will be evaluated based on **clarity of the text** and the show of **understanding of the concepts** involved.

Each identified error can receive 4 points:

- **1 point**: a brief description that shows the error was correctly identified
- **2 points**: a detailed description that shows a deep understanding of the problem and concepts involved
- **1 point**: a fix that completely solves the problem (might require changes in multiple places)

Trivial answers, brief answers, or compiler messages do not show an understanding of the concepts and receive 0 points. Examples of insufficient/incomplete answers:

- "Line `x` is not correct."
- "You are not allowed to write code like on line `x`."
- "You should replace line `x` with …"

Once you start the test, the timer begins; if you exit or get disconnected you must reconnect immediately not to lose time. If you remain disconnected, your test will be submitted after the timer reaches the end of the duration of the test.

Plagiarism

You are allowed to use any tool and material available to you during the test. The following are considered plagiarism:

- exchanging messages in any form with another person during the test.
- allowing access to your questions/solutions to somebody else before the test due date.
- acquiring the test questions/solutions that somebody else had before the test due date.

Suggestions

Look carefully at every line in the code snippet you are provided. Ask yourself *“For this line to work, what conditions must be met?”*, and then check that those conditions are met. Example of conditions (the list below is general and not complete, just gives you some examples):

- identifiers are declared/defined
- necessary headers are included
- types match in the expression
- keywords are used with the correct semantic
- operators and language features are used following the C++ standard
- correct namespaces are used
- null terminator for strings is used appropriately
- privacy rules are not violated
- variables are not used before being initialized
- dynamic memory is correctly handled
- inheritance/polymorphism rules are followed correctly
- templates use correct template parameters when instantiated

Question 1

Needs Grading

The code below contains **five** (5) syntactic errors under C++17 standard (errors that are caught by a compiler or generate crashes/undefined behaviour at runtime).

```
// A.h
#ifndef A_H
#define A_H

struct A
{
public:
    double m_val;
public:
    A operator+=(const A& other)
    {
        this->m_val += other.m_val;
        return *this;
    }
    double getValue() const { return m_val; }
};

decltype(A().getValue()) operator+=(double& val, const A& other);

// "data" is an array of "N" elements of type "T"
template <typename T, double N>
T process(const T* data)
{
    T sum{};
    for (const auto& elem : data)
        sum += elem;
    return sum;
}
#endif
```

```
// A.cpp
#include "A.h"

decltype(A().getValue()) operator+=(double& val, const A& other)
{
    return val += other.getValue();
}
```

```
// main.cpp
#include <iostream>
#include "A.h"
using namespace std;

int main()
{
    int arrI[5]{ 0, 1, 2, 3, 4, 5 };
    cout << process(arrI) << endl;

    A arrA[5]{ {1.2}, {2.3}, {3.4}, {4.5} };
    cout << process(arrA) << endl;
13. }
```

Your task is to identify each one by the file name and line number and **explain why the error appears, what C++ standard rule is broken, what C++ feature is misused and how the error should be fixed**. Your answer will be evaluated based on **clarity of the text** and the show of **understanding of the concepts** that are involved in the error.

Write in the answer box your solution, using the following template:

```
Error 1:



Error 2:



Error 3:



Error 4:



Error 5:
```

← **OK**