CS205 C/C++ Program Design (S22) Assignment Grading Policy

Mar. 1, 2022

This is a C/C++ assignment rules, we will deduct scores according to this rule, but we will not strictly follow it all the time. In some cases, we may decide to detract more or less points based on the perceived gravity of the mistakes. [1]

In C/C++ assignments, You are required to write **code and reports** and submit them to blackboard . We provide the standard of how we judge your assignments. There are several rules you need to follow.

1 Standard

Standard	Score
Report	20
Output correctness & Error handling	70
Code style	10

1.1 Notice

1. We strongly recommend you to **submit your assignments before deadline**. We will **deduct 20 scores each day** if you submit assignment after deadline. If you did not submit your assignment three days after deadline. You will get 0 for your assignment.
2. Please submit `.pdf` file for your report. If you do not submit the report,we will **deduct 20 scores**; if you submit other types file such as `.doc` , `.docx` ,we will **deduct 10 scores** from this part.
3. Please submit all the required source code. If you do not submit source code *as specified in each assignment*, which makes your program cannot be run properly, you will be **deducted up to 80 scores**.
4. **No plagiarism.** If we find that you have plagiarized other's code or take the code from some websites strightly without citation, you will get 0 scores for this assignment ; if we find that you have plagiarized more than once, you will get 0 scores for this course.

Suspects on both sides of the plagiarism will be punished.

5. The scores marked for each task in the document are raw scores. Your final score is

$$\min(\Sigma_{task}\ score * 0.7 + score_{report} + score_{code\ style}, 100)$$

6. You may notice that in some assignments, $\Sigma_{task}\ score > 100$, which means: [i] you may skip several tasks (but leaving a blank function body); [ii] you may take the excess of 100 pts as a bonus and use it to offset other possible deductions.

2 Rules for score deduction

2.1 Compile and Run

1. Please specify your develop environment *especially when you are using some platform-dependent APIs* (e.g. `windows.h`). TA will try his best to run your code.

> 1. *Please include all the special of compilation environment like **compilers**, **libc** or **operating system** in your **report**.*
> 2. *For Windows users, if you are **not** using WSL, you may using `MSVC` / `mingw`; for WSL / Linus users, you may using `gcc` / `g++`; for macOS users, you may using `Apple clang + LLVM` by default. If you are not sure about this, you may contact SAs.*
> 3. *We will use `C++20` or `C17` standard to run your code. If you use a newer standard, like `C++23`, please also specify that.*

2. A program that does not compile will not get any points.

2.2 Output correctness

1. We will use some of the test cases to test the correctness of your program.
2. Each test case value the same score.
3. We will provide some of the test cases to you, making sure that you can eventually get scores (~50-60% in code part) from these test cases as long as you spend enough time in the assignment.

2.3 Error handling

1. Please make sure your code will not crash if the input is incorrect. Your code should give some notice when the input is incorrect. All the required error handling will be noticed in the assignment document.
2. We will use some of the test cases to test the error handling of your program.
3. If any **memory leak** is found, your score of the `code` part will be deducted based on the perceived gravity of the mistakes.
4. If there are scoring rules for some tasks in the assignment document, the document will prevail.

2.4 Code style

Please pay more attention to your code style. After all this is not ACM-ICPC contest. You have enough time to write code with both correct result and good code style. You will get deduction if your code style is terrible. You can read Google C++ Style Guide, NASA C Style Guide or some other guide for code style. Here are some bad examples:

```
void gao();  // bad function name
int a, aa, aaa;  // bad variable name
void main(){

}  // bad main function

              vector<int> vec;
      vec.push_back(1);
        for(auto& element:vec)
    std::cout<<element<<std::endl;
// bad spacing
```

[1] This rule will take effect from the date of publication. ↩