1 Assignment 1 Introduction

This is the first graded assignment. The points are equally distributed between the two coding problems. This assignment is due *before* 11:59pm on the listed date. Which means, submissions made *on or after* 11:59pm will be counted as a late submission. Late submissions *before* 11:59 pm on the following day will receive a 25% point deduction as a penalty. Submissions made *on or after* 11:59pm on the same day will *not* be graded. We strongly recommend completing the assignment before then.

Assignments due on Mondays generally involve materials covered in lectures from the previous week, whereas assignments due on Thursdays involve materials covered in lectures from the running week. So, be sure to watch the lectures and go over the reading materials before attempting the assignments.

Good luck!

2 Wrong Answers Only

You were initially tasked with writing a program that takes two inputs from the user, and outputs their sum. So, you wrote the following program:

```
int ComputeSum(int first_num, int second_num){
    // Your code here.
}

int main() {
    // Your code here.
    return 0;
}
```

When executed, your program behaves as expected. For example:

```
Enter first number: 5
Enter second number: 13
Your sum: 18
```

However, your course instructor accidentally let slip that he lost his test script for this particular problem, and therefore, will not be able to verify your answers.

You are not the kind of student to pass up on an opportunity like this. You decide to write a program that *always* outputs the wrong answer.

General Instructions:

- When writing your program, you may only use concepts that you have learned in the course thus far.
- Your program should compile and run. You can assume that the inputs will always be integers.
- Your program should *always* (for this problem only) output wrong integers.
- The input and output prompts (not results) must match exactly as shown in the example above, including whitespaces.

3 Assignment Grade Calculator

You are to write a program that keeps track of your progress in this course. For now, you just need your program to be able to take two assignment names, along with their corresponding grades as input, and output the average.

Example Input:

```
Assignment 1 100 Assignment 2 95.5
```

Expected Output:

```
The average of your Assignment 1 and Assignment 2 grades is 97.75.
```

General Instructions:

- When writing your program, you may only use concepts that you have learned in the course thus far.
- Your program should compile and run. You can assume that the assignment grades will always be between 0 and 100.
- Your program must make use of a *ComputeAverage* function. The purpose of this function should be self-explanatory.
- Your program should take one single sentence (6 words separated by 5 whitespaces), and separate the input according to the program specification. You are *not* allowed to take the inputs for two assignments separately. In fact, if your solution uses more than one *std::cin*, then you are probably doing something wrong. You do not need to do any string manipulation to solve this problem. Think about how you can take multiple values from an input into multiple variables, when they are separated by whitespaces.
- The formatting should match exactly as shown in the example above, including white-spaces. Do not add unnecessary prompts/texts that are not specified in the problem. For example, your input should *not* have a prompt like 'Please enter your assignment names and grades: '. In fact, this particular problem should not show any input prompts at all.

4 Assignment 1 Submission Process

- Create a folder, name it *your_msu_id1*. For example, if your MSU email is *johndoe@msu.edu*, then you should name the folder *johndoe1*. For Assignment 2, it should be *johndoe2*, and so on.

- For each programming task, create a sub-folder inside your *your_msu_id1* folder, and name it as the number that corresponds to the programming task number. For this assignment, there should be two sub-folders named '2' and '3'.

- Inside each sub-folder, put the *main.cpp* for the appropriate solution.

- Compress/Zip *your_msu_id1* folder and name it *your_msu_id1.zip*. For example, if the name of your folder is *johndoe1*, then you need to create a zip file named *johndoe1.zip*. See here if you are not sure how to create zip files: https://copyrightservice.co.uk/reg/creating-zip-files.

- Submit the zip file through D2L.

Reference submission: https://cse232msu.github.io/assets/files/johndoe0.zip.