ASSIGNMENT: C/C++ Pointers

Start Assignment

- Due Thursday by 10pm
- Points 20
- Submitting a text entry box
- Available Feb 6 at 1:35pm - Feb 12 at 10pm

Assignment 1

C / C++ Pointers

Basic Task - to earn 15/20 points

Create a C++ program that will prompt the user for information about a student and their grades. The program should ask the user for the following information:

- User ID number (int)
- First name (string)
- Last name (string)
- Number of assignments (int)
- Grade for each of the assignments (series of doubles)

The program should then calculate the average score of the assignments and print the results (rounding the average to 1 digit after the decimal place).

Example

```
Please enter the student's id number: 123456
Please enter the student's first name: Jane
Please enter the student's last name: Doe
Please enter how many assignments were graded: 4

Please enter grade for assignment 0: 93.5
Please enter grade for assignment 1: 86.667
Please enter grade for assignment 2: 100
Please enter grade for assignment 3: 91.25

Student: Jane Doe [123456]
  Average grade: 92.9
```

Note: computer output is shown in black, and user input is shown in blue.

Instructions

Fork **https://github.com/tmarrinan/os-pointers** (https://github.com/tmarrinan/os-pointers) to create your own copy of the os-pointers git repository. Then clone (do not simply download) your repository.

Use the provided pointers.cpp file as a starting template for this assignment. You must use the provided functions (cannot just define a new function to calculate the average, or calculate it directly in main). Your output format should match that of the example above, including whitespace. Note that input grades can have many digits after the decimal point, but the average grade should be rounded to just 1 digit.

ASSUMPTIONS: you can assume the following (i.e. no test case will violate these assumptions)

- Student IDs will have not more than 9 digits
- Names will be less than 128 characters long
- Names will not have any whitespace
- Names will not contain non-ASCII characters
- Number of assignments is less than 134,217,728
- Grade scores will be less than 1000.0, and not contain more than 6 digits after the decimal (i.e. max number of characters is 10 including the decimal point)

Advanced Task - to earn 20/20 points

Error check that the user is inputting the right data type. If not, the program should print "Sorry, I cannot understand your answer", then re-prompt the user for the same field. Note that both the student ID and all grades should be limited to non-negative numbers. The number of assignments should be limited to numbers >= 1.

For example:

```
Please enter the student's id number: 123456
Please enter the student's first name: Jane
Please enter the student's last name: Doe
Please enter how many assignments were graded: A6
Sorry, I cannot understand your answer
Please enter how many assignments were graded: 4x
Sorry, I cannot understand your answer
Please enter how many assignments were graded: 2

Please enter grade for assignment 0: 98
Please enter grade for assignment 1: -10
Sorry, I cannot understand your answer
Please enter grade for assignment 1: 92.6

Student: Jane Doe [123456]
  Average grade: 95.3
```

Testing

There is a sub folder with 2 test cases - each containing input and its expected output. In order to test your code against expected output, you can do the following:

- Run your program, retrieving standard input from a file (e.g. 'input.txt') and outputting standard output to a separate file (e.g. 'output.txt')

    -
      ```
      ./pointers < input.txt > output.txt
      ```
- Compare output from your program to the expected output (e.g. 'expected.txt')

    -
      ```
      diff --strip-trailing-cr output.txt expected.txt
      ```
    - The `diff` command outputs differences between the two files. If the two files are identical, then there will be no output

IMPORTANT: Your code should accept input and produce output *exactly* like the examples above. I will be running your code through a number of test cases (beyond the two provided) and checking the difference between your program and a correct solution. Therefore even printing numbers with extra digits after the decimal place or using an extra white space in the output could cause a test case to fail.

Submission

Submit the projects GitHub URL along with compilation and running instruction in Canvas.

Deadline

This assignment is due Thursday, February 12 at 10:00pm.