You must submit a zipped, **and only zipped**, archive of your project directory, `exam1.zip`. The directory should contain, at a minimum, the files

1. `problem1.cc`
2. `problem2.cc` and `problem2.h`
3. `problem3.cc` and `problem3.h`

in a directory named `exam1`.

A different makefile will be used to make your project files.

You are provided a makefile that will build and test all problems. If you are working from a BASH shell (it is likely called wsl if you are running WSL on windows), try typing

make problem

followed by the tab key a couple times. This will show you all possible build options for each problem, including style tests.

Note this is an exam. All code submissions will be carefully considered for plagiarism and anyone found in violation will be remitted to Academic Integrity. In addition to comparison between individuals, I also check the larger GroupMe groups and Academic "help" websites.

You may not ask for help on the exam from SI. He has been given the exam and will not answer questions pertaining directly to it. If he feels that you are pushing his boundaries, he has been instructed to refer your name to me and it will be held as a violation of the Carolina Creed.

That said, you are free to ask him general concept questions. We have spoke at length and he believes he can answer general questions without violating the integrity of the examination.

Problem1: infix notation floating point comparator

Task: WITHOUT PROMPTING read two numbers and an operator from STDIN, using std::cin three times or in its form:

```
std::cin >> value1 >> value2 >> value3;
```

The input will always be given as

number operator number

e.g., 2.3 == 2.3, 4.5 != 4.5, etc.

Using the operands and operators and conditional statement(s) perform the operation. Emit the string "true" or "false" as necessary.

DO NOT emit anything other than the result. This is a calculator. Simply emit the result of the operation. Do not prompt for input. Just write the result of the operation to STDOUT using std::cout.

The two operands can be read as floating point data and the operator can be read as an std::string using the std::cin object. Comparison is well-defined between std::string instances, so you are safe to do so. You need not set precision for the floating point (in)equality operators. The default precision for a double is all that is required.

You must provide operators:

<

<=

==

>=

>

! =

Points:

compilation: 1

style: 1

correctness: 2

Cont.

Problem 2: SumDigits

The function accepts a read-only std::string reference containing a signed integer value and returns the sum of its digits as a string. For example, input “12345” returns “15”, i.e. 1 + 2 + 3 + 4 + 5. Negative values should be expected, such that “-12345” returns “-15”.

Note that if you solve the problem by conversion to integers and back, the functions std::stoi and std::to_string are useful and are found in the string library already included. Usage of integer modulus by 10 (int_val % 10) and integer division by 10 (int_val / 10) can extract the ones-place digit and shift-right all digits to the left of the ones-place.

A different solution would see you converting each character in the string value to an integer and summing them. You would need to handle (possible) leading signs, e.g., 12, +12, -12, etc.

Sometimes there may be errors such that strings containing something other than integers are passed in. You should detect these errors and return the empty string (“”), e.g., 1two3, +twelve, etc. If you use std::stoi, reference the method you used in homework program 2, if not you will need to devise your own.

Points:

compilation: 1

style: 1

correctness: 3 (non-negative, negative, error)

Cont.

Problem 3: File Processing

These functions accept a string file name, open the file, read the contents, and processes the contained data as indicated (excluding the count of doubles).

The file format consists of an integer n, followed by n additional positive and negative doubles. The first value is not part of the dataset. The values are all separated by white space and so can be parsed using the insertion operator (>>). You should probably use the fstream object (which accepts a string in its constructor) to read the doubles.

Functions:

- double Max(const string&):
  Accepts string parameter as file name, returns double value representing the largest value of the doubles in file
- double Min(const string&):
  Accepts string parameter as file name, returns double value representing the smallest value of the doubles in file
- double Sum(const string&):
  Accepts string parameter as file name, returns double value representing the sum of the values of the doubles in file
- double Avg(const string&):
  Accepts string parameter as file name, returns floating point value representing the sum of the values of the doubles in file
- void Sort(const string&, int[]):
  Accepts string parameter as file name and an double array parameter to store the values in the file pre and post sort. The double array will be large enough to hold all double values (except count) in the file; this is a precondition. The function stores the sorted values from the file in the double array. Notice that this means you must use an in-place sort to sort the array. YOU MAY NOT use the algorithm library. Any student doing so will receive 0/6 for this problem.

References:

fstream: http://cplusplus.com/reference/fstream/

string: http://cplusplus.com/reference/string/

Points:

compilation: 1

style: 1

correctness: 4 (Max: 0.5, Min: 0.5, Sum: 0.5, Avg: 1, Sort: 1.5)

The End.