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

1. `problem1.cc`
2. `problem2.cc`
3. `problem3.cc` and `problem3.h`
4. `problem4.cc` and `problem4.h`
5. `problem5.cc` and `problem5.h`

in a directory named `exam1`.

I will use my own makefile to make your project files.

Problem 1: Prefix Notation Arithmetic Calculator

Arithmetic is customarily represented using infix notation, e.g. 1.0 + 2.0. In this assignment, you are creating a prefix notation calculator, e.g. + 1.0 2.0.

The program will read the operator and operands from the standard input stream (using cin). The operator will be first and the left-hand side and right-hand side operands will be second and third, respectively. You will then write the output of the operation to the standard output stream (using cout).

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.

You must provide operators:

+ : addition

- : subtraction

x : multiplication

/ : division

<: less than

>: greater than

Notes:

The two operands should be read as floating point data and the operator is a character. You need not set precision for the floating point output. The default precision for a double is all that is required.

Points:

compilation: 1

style: 1

correctness: 3

Problem 2: Prefix Notation Arithmetic Calculator

Arithmetic is customarily represented using infix notation, e.g. 1 + 2. In this assignment, you are creating a prefix notation calculator, e.g. + 1 2.

Read the three arguments passed into the file. They will be at indices 1, 2, and 3. The operator will be at index 1, the left and right operands at indices 2 and 3, respectively. You will then write the output of the operation. DO NOT emit anything else. This is a calculator. Simply write the result of the operation to the standard output stream (using cout).

You must provide operators:

+ : addition

- : subtraction

x : multiplication

/ : division

% : modulo

ˆ : exponentiation

Notes:

The two operands should be converted to integral data and the operator is a character. The arguments are passed in as character arrays (char* argv[]) and so the operator cannot be directly compared to string literals such as "+". I would recommend using the string class, which accepts a character array in its constructor and provides the == (equivalency) operator to determine which operator was provided.

The operands can be converted from character arrays (char* argv[]) to signed ints by the atoi function.

References:

atoi: http://www.cplusplus.com/reference/cstdlib/atoi/

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

Points:

compilation: 1

style: 1

correctness: 3

Cont.

Problem 3: SumDigits

The function accepts a signed integer value and returns the sum of its digits. For example, input 12345 returns 15, i.e. 1 + 2 + 3 + 4 + 5. Negative values should be expected, such that -12345 returns -15.

Points:

compilation: 1

style: 2

correctness: 2

Problem 4: DecimalToBinary

Accepts an unsigned integer value and returns the binary representation of that number. For example, 3 returns the string “11”, 5 returns the string “101”, and 17 returns “10001”.

A simple algorithm you might use is as follows:

1. Begin with an empty string.
2. Divide decimal value by 2. If there is a remainder, add a ”0” to the beginning of the string or a ”1” if not.
3. Reduce the decimal value by half, taking its floor, e.g. floor(5 / 2) = 2.[1]
4. Repeat steps 2 and 3 until the decimal value is 0.

References:

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

Points:

compilation: 1

style: 2

correctness: 2

[1] Note this is the behavior of integer arithmetic.

Problem 5: LargestInteger

The function accepts a string file name, opens the file, reads the contents, and returns the largest value found inside (excluding the count of integers).

The file format consists of an integer n, followed by n additional positive and negative integers. 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 integers.

References:

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

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

Points:

compilation: 1

style: 2

correctness: 2

The End.