Computer Programming Assignment 5

Checkpoints

1. You should do the assignment in your own. You are not allowed to share code with others and/or copy code from other resources. If you are caught, as in the syllabus, you will get a failing grade.
2. We will grade using C++ g++ 8.2.1. Program failed to compile and/or run will result 0.
3. Do not loop your program to repeat unless you are told so.
4. Do not change input and output formats unless you are told so.
5. Write your name and student number at top of program as a comment.
6. Do not include Korean (and any other language than English) comments. In some encoding formats, Korean comments will cause compilation errors in the Linux environment, which will result in a 0 for your grade.
7. If you have any questions, please contact cp2018ta@tcs.snu.ac.kr.
8. Do not include “package com;” in your Java code.

Submission

1. Submit your assignment on eTL.
2. Zip your file (or tar) as ‘<Student ID>-assign5.zip’.
    a. ex.) 2017-12345-assign5.zip
3. Due date of this assignment is Dec 12th, 2018, 23:59
4. No late submission is allowed.
5. Include “Makefile” into your zip file. This “Makefile” will be used to compile C++ codes for Problem 2. Details how to write a Makefile will be given in the Lab 10.

Problem 1 Operator overloading

Given relationship of mathematical entities scalar and vector, write codes `cpScalar.hpp`, `cpVector.hpp` to perform arithmetic operations (`+`, `-`, `*`, `/`) and an `<<` operator by operator overloading.

(Scalar)
A scalar is a real number (e.g., 1, 3.14, etc.). A class `cpScalar` implementing scalar has two constructor methods:
```
cpScalar(int num)
cpScalar(double num)
```

For `<<` operator, insert `num` to the stream. This operator writes "double" type number, and truncating during calculation process should not be allowed.
ex)
//calculation of 3 + 5/4
(X) 3 + 1 = 4
(O) 3 + 1.25 = 4.25

(Vector)
A vector is a sequence of `N` scalar numbers. (e.g., [1, 2], [3, 4, 5, 6, 7], etc.). A class `cpVector` implementing vector has a single constructor method:
```
cpVector(cpScalar[] sarr, unsigned int size)
```
Number of elements in `sarr` should be identical to the variable `size`. It is an error if those are not same.

For `<<` operator, insert a beginning [, `num` of scalar 1, a comma, a space, `num` of scalar 2, …, `num` of scalar `N`, a closing] to the stream. (e.g., [], [7], [3.5, -2], etc.)
The empty Vector will not be used for any arithmetic operations.

(Operations)
Arithmetic operations between two `cpScalar`s are defined identical to those on the numerical values. The result's type will be "`cpScalar`" and you can assume only two Scalar or Vector will be considered.
Arithmetic operations between two `cpVector`s (with same size) are defined as follows:
`+`: element-wise scalar `+`. [3, 5] + [2, -1] is [5, 4]. (return type: `cpVector`)
`-`: element-wise scalar `-`. [7, 0, 2] - [3, -2, 1] is [4, 2, 1]. (return type: `cpVector`)
`*`: dot product, defined as $\sum_{i=1}^{N} v_i * v'_i$. [9, -2] * [0, -1] is scalar 2. (return type: `cpScalar`)
`/`: divide each element of the first by the sum of absolute values of the second vector.
(return type: `cpVector`)
[4, 6] / [-1, 1] is [2, 3]. (since |-1| + |1| = 2)

[3, 5] / [2, 2] is [0.75, 1.25].

Arithmetic operations between one `cpScalar` (left) and one `cpVector` (right) are defined as follows:

`+`: scalar `+` to each element in vector. 4 + [1, -2] is [5, 2]. (return type: **`cpVector`**)

`-`: deduct the sum of the absolute values of vector elements from a scalar.
(return type: **`cpScalar`**)

4 - [1, -2] is 1. (since | 1 | + | -2 | = 3, 4 - 3 = 1)
8 - [1, -2, 3] is 2. (since |1| + |-2| + |3| = 6, 8 - 6 = 2)

`*`: scalar `*` to each element in vector. 2 * [-1, 3] is [-2, 6]. (return type: **`cpVector`**)

`/`: divide a scalar by the sum of absolute values of vector elements. (return type: **`cpScalar`**)
24 / [-1, 3] is 6. (since |-1| + |3| = 4, 24 / 4 = 6)

Arithmetic operations between one `cpVector` (left) and one `cpScalar` (right) are defined as follows:

`+`: same as above. [1, -2] + 4 is [5, 2]. (return type: `cpVector`)

`-`: scalar `-` to each element in vector. [3, 6] - 4 is [-1, 2]. (return type: `cpVector`)

`*`: same as above. [-1, 3] * 2 is [-2, 6]. (return type: `cpVector`)

`/`: scalar `/` to each element in vector. [3, 5] / 5 is [0.6, 1]. (return type: `cpVector`)
In this case, vector elements are divided by scalar value itself, not the absolute value.
[5, 5] / -5 is [-1, -1]

You should terminate the program when (a) `size` of `cpVector` is not identical to the number of elements in `sarr` or (b) two vectors with different size are given or (c) division by zero is performed, although graders will not test inputs satisfying those.

You do not need to implement your own `main()` method: graders will use their code, for example:

```
int main()
{
      cpScalar s1(10);
      cpScalar s2(5.0);
      cpScalar s_arr[];
      for(int i = 0; i < 4; i++){
        s_arr[i] = cpScalar(i+1);     //1, 2, 3, 4
      }
      cpVector v1(s_arr, 4);

      cpScalar result1 = s1 / s2;
      cpScalar result2 = s1 / v1;
      cpScalar result3 = v1*v1;
      cpScalar result4 = s2 / v1;
      cpVector result5 = v1 / s2;

      //<< operator overloading
      cout << result1 << ", "+ << result2 <<", " <<result3
            << ", "<< result4 << ", "<< result5 << endl;
}
/**
Expected output

2.0, 1, 30, 0.5, [0.2, 0.4, 0.6, 0.8]

**/
```

You may assume that the value of `N(size)` is not greater than 2048.

Submission

cpScalar.hpp, cpVector.hpp
[When you cannot separately define cpScalar and cpVector, you can put all contents of cpScalar.hpp[cpVector.hpp] to cpVector.hpp[cpScalar.hpp] and make cpScalar.hpp[cpVector.hpp] empty
]

Problem 2 Number Quiz

Think about a number quiz where only several arithmetic operators and a equals sign (‘=’) are specified but operands (numbers) are left as boxes.

[][]+[]-[][]=12

One of the solutions for this quiz could be 24+7-19.

Your program must print **at most three** possible solutions for a given number quiz. It will print “No solution” when no solution is possible.

You may assume that the right hand side of the equals sign is always an integer value.

The maximum number of boxes for a number is 5. ex) [][][][][]/[][][]=100 is allowed, [][]+[][][][][][]=800000 is not allowed.
You may assume the maximum number of “number” is at most 8 ~~32~~.
The total number of boxes in a quiz will be at most 10.

Precedence order of operators must be followed. ex) 2 + 3 * 2 = 8 (not 10)

Operators will be from the following list where precedence order is also specified.
[^(pow)(order 1), %(mod)(order 2), *(multiply)(order 3), /(divide)(order 3), +(add)(order 4), -(subtract)(order 4)]
For the same precedence order, leftmost one is executed first and only integer values are assume during calculation process. (i.e., all operators are left associative).
ex) 3 - 8 % 2^2 + 3 + 2^2*3 = 3 - 0 + 3 + 4*3 = 18.

Locations of boxes can be anywhere except for operators, equals sign, and the integer after equals sign. The number of boxes will be at least one. A digit for a box is a decimal non-negative number(0~9). When there are multiple digits, the first digit is not allowed to be a ‘0’.

ex) []2[] / 1[] = 11 (Solution is 1, 1, 1)
ex2) 8^[] - 1 = 63. (Solution is 2)
ex3) 7 * 5 - [] = 35. (Solution is 0)

Input will be given as a standard input where boxes are denoted by two brackets ‘[‘ , ‘]’ without any space between them. “ [] ”
ex) [][][][]/[][]=0
ex2) [][]*[]=30
ex3) []0-100=-80

After your program gets a number quiz, it must **print** solutions to the console, delimited by a newline or “No solution”, and then terminate itself.

ex) No solution
ex2) 10, 3
       15, 2
       30, 1
ex3) 2

Submission

Write a makefile to compile your C++ codes.
Grading for this problem will be done by typing `make` on command line in the Linux environment and typing [`./NumberQuiz`].