2018-1 Computer Programming Assignment 5

You should do the assignment on your own. You are not allowed to share codes with others and/or copy codes from other resources. If you get caught, you will lose all points from this assignment.

Grading will be done in Linux environment using GCC (GNU Compiler Collection) 5, identical to that inside the lab machines. Keep that in mind when writing code in other environments. Programs having any kinds of compile errors will receive neither compile nor test case points.

Do not change the format of input and output. Write everything, including comments, in English.

Upload your work in ETL. You should submit only a single TAR or ZIP file containing those files:
[1] Problem 1: `cpScalar.hpp`, `cpVector.hpp` (S/V in upper case!).
[2] Problem 2: `maze.cpp`.

Do not include any subdirectories or any other files inside your archive, so that we can see your source codes right after unzipping it.

Due of this assignment is 11PM on June 11th. No late submission is allowed.

If you have any questions, write an article to the Class Q&A board in ETL so that everyone can see what is going on. TAs will try to respond your questions and announce modifications of the specification promptly, if any. However, TAs will not be able to answer questions after 5PM on June 8th: it is kindly recommended to start your work as early as possible.

Problem 1
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)
```
to conform the definition.
For << operator, insert `num` to the stream.

(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)
```
to conform the definition.
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.)

(Operations)
Arithmetic operations between two `cpScalar`s perform identical to those in numerical values.
Arithmetic operations between two `cpVector`s (with same size) are defined as follows:
+: element-wise scalar +. [3, 5] + [2, -1] is [5, 4].
-: element-wise scalar -. [7, 0, 2] - [3, -2, 1] is [4, 2, 1].
*: dot product, defined as $\sum_{i=1}^{N} v_i * v'_i$. [9, -2] * [0, -1] is scalar 2.
/: not defined.
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].
-: not defined.
*: scalar * to each element in vector. 2 * [-1, 3] is [-2, 6].
/: not defined.
Arithmetic operations between one `cpVector` (left) and one `cpScalar` (right) are defined as follows:
+: same as above. [1, -2] + 4 is [5, 2].
-: scalar - to each element in vector. [3, 6] - 4 is [-1, 2].
*: same as above. [-1, 3] * 2 is [-2, 6].
/: scalar / to each element in vector. [3, 5] / 5 is [0.6, 1].

You should terminate the program when (a) a non-defined operator is used or (b) two vectors with different size are given or (c) zero-dividing 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.

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

Problem 2
Write a code `maze.cpp` that does identical work to `Maze.java` in the assignment 4.

Refer to assignment 4 specification for details.

If you plan to use any data structures mentioned in the lecture (e.g., stack, queue, binary tree, etc.), implement them on your own: do not import any relevant classes directly from external headers and/or libraries.