CSE 333 13su Exercise 10

out: Friday, July 19, 2013
due: Monday, July 22, 2013 by **9:00 am**.

Create a C++ class Vector that implements 3-D vectors.

- In file `Vector.h` declare a class Vector with the following properties:
  - The representation of a Vector should be three `double`s giving the magnitudes in the x, y, and z directions.
  - There should be a default (0-argument) constructor that initializes a Vector to (0,0,0), a constructor with 3 doubles as parameters giving initial values for the x, y, and z magnitudes, and a copy constructor.
  - If one is needed, there should be a destructor that does whatever work is needed when a Vector object is deleted. If no work is needed you may omit the destructor and rely on the default provided by the compiler.
  - The class should define assignment on vectors (u=v).
  - Operators + and - should be overloaded so that u+v and u-v return new Vectors that are the sum and difference of Vectors u and v, respectively.
  - Operator * should compute the inner product (dot product) of two Vectors. If v1=(a,b,c) and v2=(d,e,f), then v1*v2 should return the scalar value a*d+b*e+c*f.
  - Operator * should be overloaded so that if v is the Vector (a,b,c) and k is a double, then v*k should return a new Vector containing the components of v multiplied by k (i.e., (a*k,b*k,c*k)).
  - The class should define stream output so that s<<v will write Vector v to stream s as (a,b,c), i.e., a left parentheses followed by the x, y, and z components of v separated by commas, and a right parentheses. There should be no added spaces in the output

  Note that several of these functions are required to return new Vectors. This means actual Vector values, not pointers to Vectors that have been allocated elsewhere.

- In file `Vector.cc` implement this class.

- In file `ex10.cc` write a main program that demonstrates that your Vector class works properly. The output format is up to you, but it should be labeled neatly and should be concise so it can be read with pleasure, not pain.

Your code must:

- compile without errors or warnings on CSE Linux machines (lab workstations, attu, or CSE home VM)
- have no crashes, memory leaks, or memory errors on CSE linux machines
- compile with the command `g++ -Wall -g -std=gnu++11 -o ex10 *.cc`
- be pretty: the formatting, modularization, variable and function names, and so on must make us smile rather than cry. (cpplint may help identify things to check, although some of the things it warns about are fine for this exercise.)
- be robust: you should think about handling bogus input from the user, and you should handle hard-to-handle cases (if there are any) gracefully.
- have a comment at the top of each source file and the makefile with your name, student number, and CSE or UW email address.

You should submit your exercise using the assignment dropbox linked on the main course web page. You may submit the files either as separate files or in a single tar archive named ex10.tar.