Technical Homework 2

PHY410: Do problems 1 and 2.
PHY 505: Do all 3 problems.

Accept the assignment from github classroom https://classroom.github.com/a/J8sjJveu. You will then get a link to your own github area.
You should submit your code through github classroom. Submit your writeup, and a link to your github classroom area where your code is, on UBLearns.
Be sure to have a github account and link it to your assignment.

To submit your code:

```
git clone <insert the link to your code here>
mkdir Assignment2
cd Assignment2
<do your coding>
git add <insert filename here>
git commit -m"<Add a descriptive message here>"
git push origin master
```

Note: do not directly cut and paste the text in angled brackets above, you must insert your actual filenames and other relevant information.

My example is (and yours will be different):

```
git clone git@github.com:ubsuny/technical-assignment-2-rappoccio-1.git
cd technical-assignment-2-rappoccio-1
mkdir Assignment2
cd Assignment2
git add *.*
git commit -m"I hope I passed"
git push origin master
```

Please name your files"Problem1.cpp", "Problem2.cpp", “Problem3.cpp”.

Problem 1 (25 points):

(25 points) Create a program in a file called “Program1.cpp” that inputs a number from the command line, interprets it as a long integer with the “`atol`” function (defined in `<stdlib.h>`), and calculates its factorial, if the integer is less than 20. If the integer is greater than or equal to 20, the program should print a warning and return zero. This should use a function called "factorial" that computes the factorial with signature:

```
unsigned int factorial(unsigned int x);
```

Your code should also handle the case where no arguments are provided and remind the user of the syntax.

The unit tests will be (executing from your top level directory)

```
g++ Assignment2/Problem1.cpp -o Assignment2/Problem1
./Assignment2/Problem1
./Assignment2/Problem1 10
./Assignment2/Problem1 20
```

Problem 2 (25 points):

a. (15 points) Create a class template file called "`LorentzVector.h`" that implements a class template of a Lorentz vector using natural units c = 1 called "`template<class F> LorentzVector`". Create a method called "mass" that computes the invariant mass of a Lorentz vector ($m = \sqrt{t^2 - x^2 - y^2 - z^2}$). Make a specific class of type "`LorentzVector<double>`". Create two Lorentz vectors "`v1 (1,0,0,1)`" and "`v2 (1,0,0,-1)`". Write a program "`Problem2.cpp`" that will instantiate these two vectors and print them.

b. (10 points) Create an addition operator called "`operator+`" that will sum two four vectors and return the sum. Sum the vectors v1 and v2 from part a, using the same file "`Problem2.cpp`". Print the sum and its invariant mass.

The unit tests will be (executing from your top level directory)

```
g++ Assignment2/Problem2.cpp
./Assignment2/Problem2
```

Problem 3 (25 points) Grad Students only:

- Write a C++ program starting from "`ReviewCpp/ClassExample/read_points_example.cc`" renamed to "`Program3.cpp`". Create a class to represent a line (called "`Line`") that is declared in a file called "`Line.h`" and defined in a file called "`Line.cc`", with protected data members "`double m_;`" and "`double b_;`" that are the slope and y-intercept.
- Be sure to copy the "`Point`" class (both "`Point.cc`" and "`Point.h`") into your `Assignment2` directory from `ReviewCpp/ClassExample`.
- The class should have two construction operators and two overloaded "`set`" methods that can EITHER input two `Points` (where you compute m and b), OR can input the slope and intercept directly.
- The slope may be infinity, in which case set it to "`std::numeric_limits<double>::infinity()`".
- The y-intercept may be undefined, in which case set it to "nan" (or "not a number") by setting it equal to "`sqrt(-2)`".
- You are welcome to edit any code that you copy.

a. (15 points) Your main function should be defined in a file called "`Problem3.cpp`" that reads a file from the command line, which contains the following inputs (one per line). Compute lines from all pairs of inputs and print the equation of the line like "`y = m x + b`" where you substitute the values for m and b. The inputs are:
   - -1,-1
   - 1,1
   - 1,1
   - 1,1.00000000000000005
   - 1,2
   - 3,4
   - 1,8

b. (10 points) Sort the list of lines from part a in increasing order of slope, and print them all out with the same formatting as in a. Use the same function "`Problem3.cpp`".

The unit tests will be (executing from your top level directory)

```
g++ Assignment2/Line.cc Assignment2/Point.cc Assignment2/Problem3.cpp
-o Assignment2/Problem3
./Problem3 Assignment2/points.txt
```