Kerry Ojakian's CSI 32 Class
Class Work #3

General Instructions: Choose one person from the group. In their Dropbox create a folder named Class03. Put your .cpp file in this Class03 folder. Also, copy the parent class in there (your code should compile and run without error!) When done, email me indicating two things: 1) The first and last names of each group member, and 2) Whose Dropbox to look in.

The Assignment

Each group will get assigned a character. You will make a child class (in C++) of the class `RPG_Char` from our class work. In addition to doing the following, create some objects of your class and test out some of the methods (your code should compile and run!).

1. Magician: has additional (from its base class) data of a bunch of spells (each spell represented by a single string), and additional methods for learning spells and casting spells. For learning, add a method named `learn` which takes a single string as input and somehow stores this string. For casting, add a method named `cast`, which takes no inputs, and returns a string; it should return the **most recent** spell learned, and that spell should be removed from his store of spells. Also override the `damage` method to remove 3 hit points when called. Also add an `inventory` method for listing all the spells currently possessed; this method takes no inputs and prints all these spells.

2. Mathematician (a very powerful character ...): like a magacian, but rather than spells, she has problems (that is math problems to solve). There is a method to add a new problem for consideration, and there is a method to solve a problem.

   For adding a problem, add a method named `addProb` which takes a single string as input and somehow stores this string. For solving a problem, add a method named `solveProb`, which takes no inputs, and returns a string; it should return the **most recent** problem that was added, and that problem should be removed from her store of problems. Also override the `improve` method to add 2 hit points when called. Also add an `inventory` method for listing all the problems currently possessed; this method takes no inputs and prints all these problems.