Assignment #3
More OOP - "Is A" Relationship/Inheritance
Due: Sunday, 05/10/15, 11:59pm

Grading: For each programming assignment, you are graded by explaining and demoing your code to a TA. **You must demo your program BEFORE the next assignment is due**, and if you fail to do so, you will automatically lose 50 points! **Your job is to convince the TA that your program works correctly, i.e. show your TA how to use/break your program**☺

(80 pts) **Problem Statement (Idea from the MIT OpenCourseWare):**

You will implement a class called `Tool`. It should have an `int` field called `strength` and a `char` field called `type`. **You may make them either private or protected**. The `Tool` class should also contain the function void `setStrength(int)`, which sets the `strength` for the `Tool`.

Create 3 more classes called `Rock`, `Paper`, and `Scissors`, which inherit from `Tool`. Each of these classes will need a **default constructor that sets the strength to 1** and a non-default constructor which will take in an `int` used to initialize the `strength` field. The constructor should also initialize the `type` field using 'r' for Rock, 'p' for Paper, and 's' for Scissors.

These classes will also need a public function `bool fight(Tool)` that compares their strengths in the following way:

- Rock's strength is doubled (temporarily) when fighting scissors, but halved (temporarily) when fighting paper.
- In the same way, paper has the advantage against rock, and scissors against paper.
- The strength field shouldn't change in the function, which returns true if the original class wins in strength and false otherwise.

*******A design choice that returns a type other than a bool for a win, loss, or tie will not be penalized******

You may also include any extra auxiliary functions and/or fields in any of these classes.

In addition, you will create a class called `RPSGame`, which allows a **human to play the rock, paper, scissors game against the computer**. Your `RPSGame` must have two `Tool *` for the human and computer tool because you don't know the new tool they'll select with each round. The `RPSGame` should also have three `int` fields to keep track of the number of `human_wins`, `computer_wins`, and `ties`.

You can choose the strategy for the computer guesses, but it cannot be based on what the human selected for a tool in the current game!!! Example of a novice and veteran computer AI: http://www.nytimes.com/interactive/science/rock-paper-scissors.html?_r=0

After the human selects the tool for the current game, display the computer's tool, a message describing who won, the current stats for the wins and ties, and then ask the user if he/she wants to play again.

Example Play of Game:
Welcome to Rock, Paper, Scissors! Do you want to choose different strengths for the tools? (y-yes, n-no) n

Choose your tool (r-rock, p-paper, s-scissor, e-exit): r
Computer chose scissor.
You win!!!
Human wins: 1
Computer wins: 0
Ties: 0

Choose your tool (r-rock, p-paper, s-scissor, e-exit): r
Computer chose paper.

Computer wins! :-(
Human wins: 1
Computer wins: 1
Ties: 0

Choose your tool (r-rock, p-paper, s-scissor, e-exit): e

Things to consider:
- If you choose to set different strengths for the tools, then you need to prompt the user for his/her specific strength for their tool, and you will need to select a specific strength for the AI choice. ***If you selected one non-default strength for both, then you will not be penalized***
- You must have the proper constructors, destructors, and assignment operator overload for the `Tool`, `Rock`, `Paper`, `Scissor`, and `RPSGame` classes.
- Your member variables in all classes must be private or protected for encapsulation rules.
- You must have your class definitions in a .h file and your implemented classes in .cpp files.
- You must also have your main function in a play_game.cpp file, separated from the class implementations.
- Create a Makefile for you project.

Computer AI Contest:

For those interested, we will have a contest to see whose computer strategy can win the most!!! We'll play at least 50 games to see whose computer strategy wins the most. One rule for entering the contest: **You are not allowed to cheat!!!!**

(10 pts) **Program Style/Comments**
In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don't align everything on the left or put everything on one line!
http://classes.engr.oregonstate.edu/eecs/spring2015/cs162-001/162_style_guideline.pdf

```
/*************************************************
** Program: card_games.cpp
** Author: Your Name
** Date: 05/09/2015
** Description:
** Input:
** Output:
*************************************************/
```

(10 pts) Design for Assignment #3 changes/Testing

- **(5 pts) How did your design for searching for data in Assignment #3 change during implementation?**

- **(5 pts) What were the actual values from your testing? Did these match your expected values? What did you do to make sure you get the expected values?**

  Please see the template for this document: Polya_template.pdf

  You need to have a table with these headings:

Input Values	Expected Output	Did Actual Meet Expected?
t	Error message for choice 't' because that isn't an option. Reprompt for choice again.	Yes

In order to submit the files, you will be creating a bzipped tar ball. In order to do this, you will use the following command, adding all the source files to the end of the command:

```
tar -cjvf cs162_hwx_username.tar.bz2 file1 file2 file3...
```

This tar ball (replacing username with your ENGR username), and only this tar ball, will be submitted via TEACH.

****NOTE:** The easiest way to upload your program from ENGR to TEACH is to map a network drive to your home directory on ENGR. Mac or Windows, See: http://engineering.oregonstate.edu/computing/fileaccess/