CSE 341, Fall 2011, Assignment 6
Due: Thursday December 1, 11:00PM

Collaboration Rules: As usual, you can discuss high-level ideas with others but your work on the assignment should be your own. *Because such a central part of the assignment is reading and understanding the provided code, you should not discuss detailed questions about how particular lines of the provided code work.* But it is fine to discuss Ruby language features and library methods that are used in the code in addition to high-level ideas about how the code works.

Warning: Playing Tetris is fun, but playing too much can make the assignment take longer than necessary.

Overview: The Ruby code provided to you in `hw6provided.rb` implements a simple but fully functioning Tetris game (see, for example, `http://www.tetris.com/how-to-play-tetris/`). Evaluating the top-level method `runTetris` starts a game. You are going to create a second game that is Tetris with some *enhancements*, described below. Evaluating the (provided) top-level method `runMyTetris` must run your game. You should *not change* any of the provided code, but you will use subclassing (since, after all, this is the OOP portion of the course) to nonetheless *reuse* as much of the code as possible. Some code-copying may be necessary, but you should minimize it subject to the no-changing-provided-code rule.

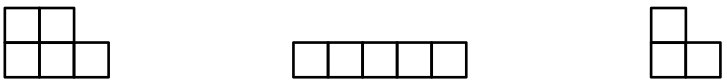
Much of the work in this assignment is understanding the provided code. There are parts you will need to understand very well and other parts you will not need to understand. Part of the challenge is figuring out which parts are which. This experience is fairly realistic.

Requirements:

- You should not change any of the provided code, nor change the definitions of any of the classes defined in the provided code (e.g., do not add or replace methods in these classes). Instead, define new classes.
- The provided Tetris game should work unchanged, with all of its features and none of the enhancements.
- Your game should have all the original features work, as well as the enhancements.
- Calling the provided top-level `runMyTetris` method must run your game.
- For simplicity (though it does not really matter), add your code to the same file as the provided code. There is a section near the bottom with the comment, "Put your solution here."

Enhancements:

1. In your game, the player can press the '`u`' key to make the piece that is falling rotate 180 degrees. (Note it is normal for this to make some pieces appear to move slightly.)
2. In your game, instead of the pieces being randomly (and uniformly) chosen from the 7 classic pieces, the pieces are randomly (and uniformly) chosen from 10 pieces. They are the classic 7 and these 3:



   The initial rotation for each piece is also chosen randomly.
3. In your game, the player can press the '`c`' key to *cheat*: If the score is less than 100, nothing happens. Else the player loses 100 points (cheating costs you) and the next piece that appears will be:

   The piece after is again chosen randomly from the 10 above (unless, of course, the player hits '`c`' while the "cheat piece" is falling).

4. **Challenge Problem:** Add a fourth enhancement to your game that is interesting and not similar to the enhancements already required. In a short text file, explain what your enhancement is and how you implemented it. Enhancements that are particularly easy will not necessarily receive credit; aim for something at least as substantial as the required enhancements and perhaps affecting a different part of the game.

   Do not share your enhancement with anyone unless they already have their own. We do not want to see the same thing twice except by coincidence.

Advice and Guidance:

- It will take some time to understand the code you are given. Be patient and work methodically. You might try changing things to see what happens, but be sure you undo any changes you make to the provided code.
- The sample solution is about 75 lines of code. As always, that is just a rough guideline; your solution might be longer or shorter.
- While you should not copy code unless necessary, it may be necessary. After all, the original game may not have functionality broken down into overridable methods the way you would want.

Turn-in Instructions

- Put your solutions in one file, `lastname_hw6.rb`, which should include the provided code. You do not need to turn in any other files unless you do the challenge problem, in which case you should have a second file `lastname_hw6_description.txt` that is an English description of what you did for the challenge problem (your code is still in the same file).
- Turn in your files using the Catalyst dropbox link on the course website.