15-112 Fall 2014 Homework 10
Due Sunday, 9-Nov, at 10pm

Read these instructions first!

- This entire hw is strictly **SOLO**, in the same way hw1-part2, hw2, and hw3 were, so you may not even discuss the problems with any other students or anyone except the course staff.
- There is no starter file this week. Submit a single file, hw10.py. Be sure to include your name and andrew id at the top.
- You may use anything at all now that is available by default with Python (without installing any additional modules or code)! OOP, recursion, events, 1d+2d lists, sets, maps, whatever.
- This week, you may only make up to 5 submissions max to Autolab. As usual, only your last one counts.
- Note: do not copy-paste the eventBasedAnimationClass.py file into your hw10.py file. Instead, include the line:
  from eventBasedAnimationClass import EventBasedAnimationClass
  Be sure that eventBasedAnimationClass.py is in the same folder as your hw10.py file when you run it. We'll do the same when we run it.
- This hw is lighter than usual, mostly to give you time to start preparing for the upcoming midterm exam, and also to give you some time to think about your term projects. Hopefully everyone will use this extra time to get in some midterm studying.

1. **5-Minute TP Meetings [5pts] [manually graded]**
   As with hw8, hw10 will include 5-minute TP meetings with one of your CA's (due by the hw10 deadline, Sunday at 10pm). This time, you need to have a bit more thinking evidenced when you show up for your meeting. This could be:

   - A list of URL's to related projects, along with your written comments on each one as to which parts of those projects are most relevant and why
   - A hand-drawn storyboard, showing in some detail what the user experience may be like for your project.
   - Some code artifacts, showing that you have made progress either getting some modules to work, or writing some prototyping code for your own project.
   - Anything else along these lines.

   The bar is low here. We do not expect major progress. But there is a bar of non-zero height -- we do expect some concrete progress. This is to keep you moving along so that when the term project is formally assigned, you are in fact ready to dive in. Also, as with the previous meetings, you need to be on time to get the points.

   This is a busy week. We know that. So don't stress over this. But do put just a bit more thought into your term project ideas, and take the steps to produce some kind of concrete deliverables (such as those outlined above) to share

with your CA in your meeting.

2. **playSameGame(rows, cols, numColors) [95pts] [manually graded]**
Read about SameGame here. You can also play an example implementation the game here.

   With this in mind, write playSameGame(rows, cols, numColors), which takes three positive integers -- the number of rows and cols of the board as well as the number of unique colors, and runs a Tkinter implementation of SameGame. You should import EventBasedAnimationClass from eventBasedAnimationClass.py (which you should not include in your submission), and then implement a class that properly extends EventBasedAnimationClass. You should also be sure you could write eventBasedAnimationClass.py from scratch, even though you will not have to do that here.

   The core game play consists of a board with cells with different colors (or cells which are empty). When the user selects a cell, you remove the region of cells connected to the selected cell which have the same color (connected meaning up, down, left, or right; no diagonal connectivity) and then "cascade" the cells which are remaining, meaning they should fall down if there are empty cells beneath them and to the right if the column to the right is empty (or, if you prefer, to the left is the left is empty; either right or left is fine, so long as you are consistent). Determining connected regions is quite similar to something we learned about in the recursion week. For cascading, you'll want to look at your Tetris implementation for inspiration.

   While you have some freedom in how exactly the game looks and works, keep in mind these requirements:

   - The board is randomly generated at the beginning of the game by picking a random color for each cell.
   - The user should select moves via the mouse (not the keyboard).
   - While clicking the mouse selects a move, hovering over a cell should highlight the region which will be removed if they click. You may highlight the region in any way you'd like that is clear, such as changing the color or shrinking the cells. Note that hovering will require that you use mouse motion events, which you will have to incorporate into the framework, including calling self.root.bind at the appropriate time and with the appropriate arguments.
   - Users cannot select a move if only one cell would be removed.
   - Cells should cascade towards the bottom right corner (or, if you prefer, towards the bottom left corner).
   - The player's score should increase based on how many cells are in the region removed (the bigger the region, the more points that should be earned).
   - The game ends if the user cannot make any more moves.
   - Pressing the "r" key should restart the game with a new randomized board.

- Your game should keep track of the highest score among multiple program invocations. To achieve this, you will need to save the high score in a file, sameGameHighScore.txt, and load the file when you start your program. If no file exists, your program should consider the high score to be None.

- Users should have 5 second to make a move. If the user makes a move, restart the 5-second timer. If the user does not make a move in 5 seconds, deduct 20pts from their score (with a minimum score of 0pts, so no negative scores) and restart the 5-second timer. If the game is over, no points should be deducted, the timer should stop, and nothing should happen until the user presses "r" to restart a new game.

- The user's score, the high score, and time remaining to make a move (as mentioned in the bullet above) should be displayed on the screen at all times.

This is not a complete spec. Resist asking for clarifications or for affirmations of your design decisions, and instead just make smart designs and fill in all the missing details. And have fun!

3. **Bonus/Optional: playSameGameBonus(rows, cols, numColors) [up to 6pts] [manually graded]**
   For bonus, be sure your playSameGame works just as noted above without modifications, but then you can also include playSameGameBonus, with the same parameters, but of course with your bonus features added. In the bonus game, you can also elect not to follow everything above strictly to the letter, if it conflicts with your new bonus features. Up to 6 points of bonus will be awarded for taking this in interesting directions. The first few points will be fairly easy to earn with some smaller but still clever features, and then the next few points will require something more clever or distinctive or algorithmically challenging. Please keep in mind that this is still only 6 points total, so do not pour endless hours into this (really!). Just have some creative fun if you have the time and inclination.