The Game of Quintris

The game of Quintris will likely seem familiar. It starts off with a blank board. One by one, random pieces (each consisting of 5 blocks arranged in different shapes) fall from the top of the board to the bottom. As each piece falls, the player can change the shape by rotating it or flipping it horizontally, and can change its position by moving it left or right. It stops whenever it hits the ground or another fallen piece. If the piece completes an entire row, the row disappears and the player receives a point. The goal is for the player to score as many points before the board fills up.

We've prepared a simple implementation of Quintris that you can play from the command line! To try it, run:

```
python3 ./quintris.py human animated
```

To play, use the `b` and `m` keys to move the falling piece left and right, the `n` key to rotate, the `h` key to flip, and the space bar to cause the piece to fall. (This works on the SICE Linux machines and should work on Mac OS command line. It may or may not work on Windows or other platforms because of the way it handles keyboard input. If it doesn't work for you, you can use a simpler but less-fun interface by running `python3 ./quintris.py human simple`.)

Your goal is to write a computer player for this game that scores as high as possible. We've provided a really simple automatic player that can be run using the command line options `computer animated` and `computer simple`. The code for this is in `tetris.py`, whereas the other python files contain the "back end" code that runs the game. You should not modify the other source code files.

We'd recommend starting with the simple version as opposed to the animated version. In the simple version, your program issues a sequence of commands which are then executed immediately before the piece begins to fall. This is simpler but prevents your program from making complicated moves (like moving left and right as the piece falls to try to wedge a piece into an awkward spot in the board). Then, consider the animated version, which also introduces an implicit time limit (since your decisions have to be made before the piece reaches the bottom of the board.)

This version of Tetris has one twist which you may want to use to get as high a score as possible: the falling pieces are not chosen uniformly at random but based on some distribution which your program is not allowed to know ahead of time. However, it may be able to estimate the distribution as it plays, which may let it make better decisions over time.

The tournament. To make things more interesting, we will hold a competition among all submitted solutions to see whose solution can score the highest across several different games. While the majority of your grade will be on correctness, programming style, quality of answers given in comments, etc., a small portion may

be based on how well your code performs in the tournaments, with particularly well-performing programs eligible for prizes including extra credit points.