CIT 594 Assignment 12: Coin Grabber (Scala)

Spring 2015, David Matuszek

Purposes of this assignment

- Get you started with the Scala programming language
- Teach you about minimaxing and alpha-beta searches

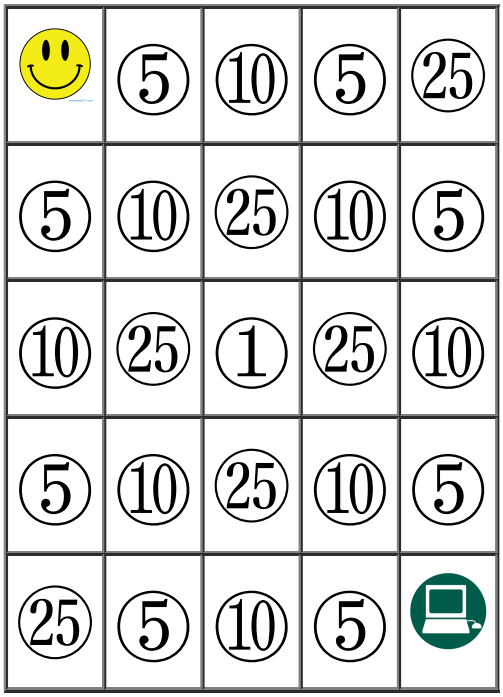
General idea of the assignment

- Write a human vs. computer game (text only, no fancy graphics)
- Use minimaxing to guide the computer's play

In more detail

Because part of the goal is to teach minimaxing, I wanted a game to which this technique could be applied. Because this is a first Scala assignment, I wanted to find a very simple game to implement. I considered Mancala, but decided against it. In the end, I just designed a game for this purpose, which I call Coin Grabber. It probably isn't going to win any "best game" prizes, though.

The game is played on a 5x5 board, arranged like this:



Here are the rules:

- Players alternate moves. The program asks the user who gets to move first.
- Players can move left, right, up, or down, but not diagonally.
- A player may not move onto a square containing the other player.
- When a player moves onto a square, he/she takes the coin on that square (if any), leaving it empty (0).

- The goal is to collect more money than your opponent.
- The game ends when either player takes the ① coin in the center of the board. At that point,
    - If the player that takes the ① coin has less than $1.00, that player immediately loses.
    - Otherwise, the player with the most money wins. (Ties are impossible.)

The human player plays by entering one of the letters `U`, `D`, `L`, or `R`, or their lowercase equivalents, or any word beginning with one of those letters (such as `down` or `duck`). The computer player indicates its move by printing one of the words `up`, `down`, `left`, or `right`. The board is printed at the beginning and end of the game, and after each computer move.

The goal is to write a program that uses a minimaxing strategy to beat, or at least challenge, the human player. Use an easily-modified constant to set the amount of lookahead. The use of alpha-beta cutoffs is optional. The program should not take more than 10 seconds to make a move (on your computer; it's okay if it takes a *bit* more time on our computers).

Most of the details are up to you. There is an obvious quantity on which to minimax--your money minus your opponent's money--but that isn't necessarily the best measure to use.

Finally: If you would prefer to implement a game of Mancala (using minimaxing), that's okay, too. Since Mancala has many variations, be sure to specify (by a `readme.txt` file or a link to a website) exactly which variation you are implementing, and make sure we understand how to run the game and make moves.

Testing

Testing is always a good idea, but I don't want to try to squeeze any more into this assignment, so unit testing is not required. If you want to test parts of your program, JUnit tests (yes, written in Java!) can be used. Or if you are really ambitious, you could look into ScalaTest (but be prepared to be overwhelmed by the options).

Grading

Grading will *probably* be as follows, but is subject to change:

- 80 points for a working program.
- 10 points if the program chooses its move within the time limit.
- 10 points if, in our subjective judgment, the game plays well enough to challenge a human.
- 10 bonus points if we can't beat your program.

Due date

Turn your assignment in to Canvas before **6 a.m. Tuesday, April 21**.