```
'''
Created on Mar 1, 2015

@author: mroch
'''

import platform   # operating system platform

import checkerboard
import charIO  # character IO

import abstractstrategy

class Strategy(abstractstrategy.Strategy):
    "Human player"

    def play(self, board, hints=True):
        """play - make a move
        Given a board, find a move and return a tuple of the new board
        and the action that created it.  If no moves are possible or the
        player forfeits, return the original board and the empty move []\

        Hints provides the user with a list of possible moves to choose
        from and is currently the only way to use this function.
        If you have problems with the unbuffered read charIO.getch(),
        use charIO.getchBuffered() which reads input in a buffered
        manner (you must press carriage return).  The charIO module has
        only been tested on CentOS 6.6 linux and Windows.
        """
        actions = board.get_actions(self.maxplayer)

        forfeit = "F"  # Human choice for forfeiting

        if actions:
            if hints:
                print(board)  # Show player current board

                # Show actions labeled a, b, c, etc.
                letter_a = ord('a')  # get encoding for "a"
                letters = [chr(letter_a + x) for x in range(len(actions))]
                for (action, letter) in zip(actions, letters):
                    print("%s: " % (letter), end=' ')
                    print(action)

                # Read the players choice and convert to action
                print("%s move, choose by letter or F to forfeit: "%(self.maxplayer), end=' ')
                letters.append(forfeit)
                choice = charIO.getch()
                print(choice)
                while choice not in letters:
                    choice = charIO.getch()

                # Pick action (None if weak-minded human forfeited)
                action = actions[ord(choice)-letter_a] if choice != forfeit else None

            else:
                raise NotImplementedError(" ".join([
                    "Write an input routine/GUI if you have too much",
                    "time on your hands.  Be sure to verify that the",
```

```
60                      "resulting action is in the list of actions"]))
61          else:
62              action = []  # No possible actions
63
64          # Execute human move
65          if not action:
66              newboard = board
67          else:
68              newboard = board.move(action)
69
70          return (newboard, action)
71
72      def utility(self, state):
73          pass  # Use gray matter human...
74
75
```