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

@author: mroch
'''
import copy
import checkerboard

boards = dict()

def init_boards():
    """Set up a library of board positions for test purposes
    WARNING:  Some components of the checkers program rely on this library
    for testing.  Changing board configurations will result in breakage of
    tests if the tests are not updated.  Adding new tests is fine.
    """


    # Initial board
    boards["Pristine"] = checkerboard.CheckerBoard()

    # Set up for two red single hops
    #           0  1  2  3  4  5  6  7
    #        0  .  b  .  b  .  b  .  b
    #        1  b  .  b  .  b  .  b  .
    #        2  .  b  .  .  .  .  .  b
    #        3  .  .  .  .  .  .  b  .
    #        4  .  .  .  b  .  .  .  r
    #        5  r  .  r  .  r  .  .  .
    #        6  .  r  .  r  .  r  .  r
    #        7  r  .  r  .  r  .  r  .
    b = checkerboard.CheckerBoard()
    b.place(2, 3, None)
    b.place(2, 5, None)
    b.place(3, 6, 'b')
    b.place(4, 3, 'b')
    b.place(5, 6, None)
    b.place(4, 7, 'r')
    b.recount_pieces()  # Update pawn/king counts

    boards["SingleHopsRed"] = b

    # Set up for black single hops
    #     0  1  2  3  4  5  6  7
    #  0  .  b  .  b  .  b  .  b
    #  1  b  .  b  .  b  .  b  .
    #  2  .  b  .  .  .  .  .  b
    #  3  .  .  .  .  .  .  r  .
    #  4  .  .  .  b  .  .  .  r
    #  5  r  .  r  .  r  .  .  .
    #  6  .  .  .  r  .  r  .  r
    #  7  r  .  r  .  r  .  r  .
    b = copy.deepcopy(b)
    b.place(6, 1, None)
    b.place(3, 6, 'r')
    b.recount_pieces()  # Update pawn/king counts
    boards["SingleHopsBlack"] = b

    # multihop
```

```
60      #     0  1  2  3  4  5  6  7
61      #  0  .  b  .  b  .  b  .  b
62      #  1  b  .  r  .  b  .  .  .
63      #  2  .  r  .  .  .  b  .  b
64      #  3  .  .  .  .  .  .  .  .
65      #  4  .  .  .  r  .  b  .  .
66      #  5  .  .  .  .  .  .  r  .
67      #  6  .  r  .  r  .  r  .  r
68      #  7  r  .  .  .  r  .  .  .
69      b = checkerboard.CheckerBoard()
70      b.place(7, 2, None)
71      b.place(7, 6, None)
72      b.place(5, 0, None)
73      b.place(5, 2, None)
74      b.place(5, 4, None)
75      b.place(4, 3, 'r')
76      b.place(4, 5, 'b')
77      b.place(2, 1, 'r')
78      b.place(2, 3, None)
79      b.place(1, 2, 'r')
80      b.place(1, 6, None)
81      b.recount_pieces()  # Update pawn/king counts
82      boards["multihop"] = b
83
84
85      # KingBlack
86      # Black can move to become a King but should
87      # not be able to move after being kinged
88      #    0  1  2  3  4  5  6  7
89      #    0  .  .  .  .  .  .  .  .
90      #    1  .  .  .  .  .  .  .  .
91      #    2  .  .  .  .  .  .  .  .
92      #    3  .  .  .  .  b  .  .  .
93      #    4  .  .  .  r  .  r  .  .
94      #    5  .  .  .  .  .  .  .  .
95      #    6  .  .  .  r  .  r  .  .
96      #    7  .  .  .  .  .  .  .  .
97      b  = checkerboard.CheckerBoard()
98      # clear the board
99      for r in range(b.rows):
100         for c in range(b.coloffset[r], b.cols, 2):
101             b.place(r, c, None)
102     # Set up for tour by black
103     b.place(3, 4, 'b')  # pawn that will be making partial tour
104     b.place(4, 3, 'r')
105     b.place(6, 3, 'r')  # king black after this jump
106     b.place(6, 5, 'r')  # or this one depending on path
107     b.place(6, 5, 'r')
108     b.place(4, 5, 'r')
109     b.recount_pieces()  # Update pawn/king counts
110     boards["KingBlack"] = b
111
112     # BlackKingTour
113     #    0  1  2  3  4  5  6  7
114     #    0  .  .  .  .  .  .  .  .
115     #    1  .  .  .  .  .  .  .  .
116     #    2  .  .  .  .  .  .  .  .
117     #    3  .  .  .  .  B  .  .  .
118     #    4  .  .  .  r  .  r  .  .
```

```
119     #    5  .  .  .  .  .  .  .  .
120     #    6  .  .  .  r  .  r  .  .
121     #    7  .  .  .  .  .  .  .  .
122     b = copy.deepcopy(b)
123     b.place(3, 4, 'B')  # king that will make tour
124     b.recount_pieces()
125     boards["BlackKingTour"] = b
126
127     # RedKingTour
128     # Probably don't need to test this one as rules similar, but...
129     #    0  1  2  3  4  5  6  7
130     #    0  .  .  .  .  .  .  .  .
131     #    1  .  .  .  .  .  .  .  .
132     #    2  .  .  .  .  .  .  .  .
133     #    3  .  .  .  .  R  .  .  .
134     #    4  .  .  .  b  .  b  .  .
135     #    5  .  .  .  .  .  .  .  .
136     #    6  .  .  .  b  .  b  .  .
137     #    7  .  .  .  .  .  .  .  .
138     b = copy.deepcopy(b)
139     b.place(3, 4, 'R')  # pawn that will be making partial tour
140     b.place(4, 3, 'b')
141     b.place(6, 3, 'b')  # king red after this jump
142     b.place(6, 5, 'b')  # or this one depending on path
143     b.place(6, 5, 'b')
144     b.place(4, 5, 'b')
145     b.recount_pieces()
146     boards["RedKingTour"] = b
147
148     b = checkerboard.CheckerBoard()
149     b.clearboard()
150     b.place(0, 1, 'b')
151     b.place(0, 7, 'b')
152     b.place(1, 6, 'r')
153     b.place(2, 1, 'r')
154     b.place(2, 3, 'b')
155     b.place(2, 5, 'b')
156     b.place(4, 3, 'r')
157     b.place(5, 4, 'r')
158     b.place(6, 1, 'b')
159     b.place(6, 3, 'r')
160     b.recount_pieces()
161     boards["StrategyTest1"] = b
162
163     # EndGame 1 - Red can easily win
164     #       0  1  2  3  4  5  6  7
165     #    0     .     .     R     b
166     #    1  .     .     .     .
167     #    2     .     .     .     .
168     #    3  .     .     .     .
169     #    4     .     .     .     .
170     #    5  .     .     .     .
171     #    6     .     .     .     R
172     #    7  .     .     .     .
173     b = checkerboard.CheckerBoard()
174     b.clearboard()
175     b.place(6,7, 'R')
176     b.place(0,5, 'R')
177     b.place(0,7, 'b')
```

```
178     b.recount_pieces()
179     boards["EndGame1"] = b
180
181
182 init_boards()
183
184
```