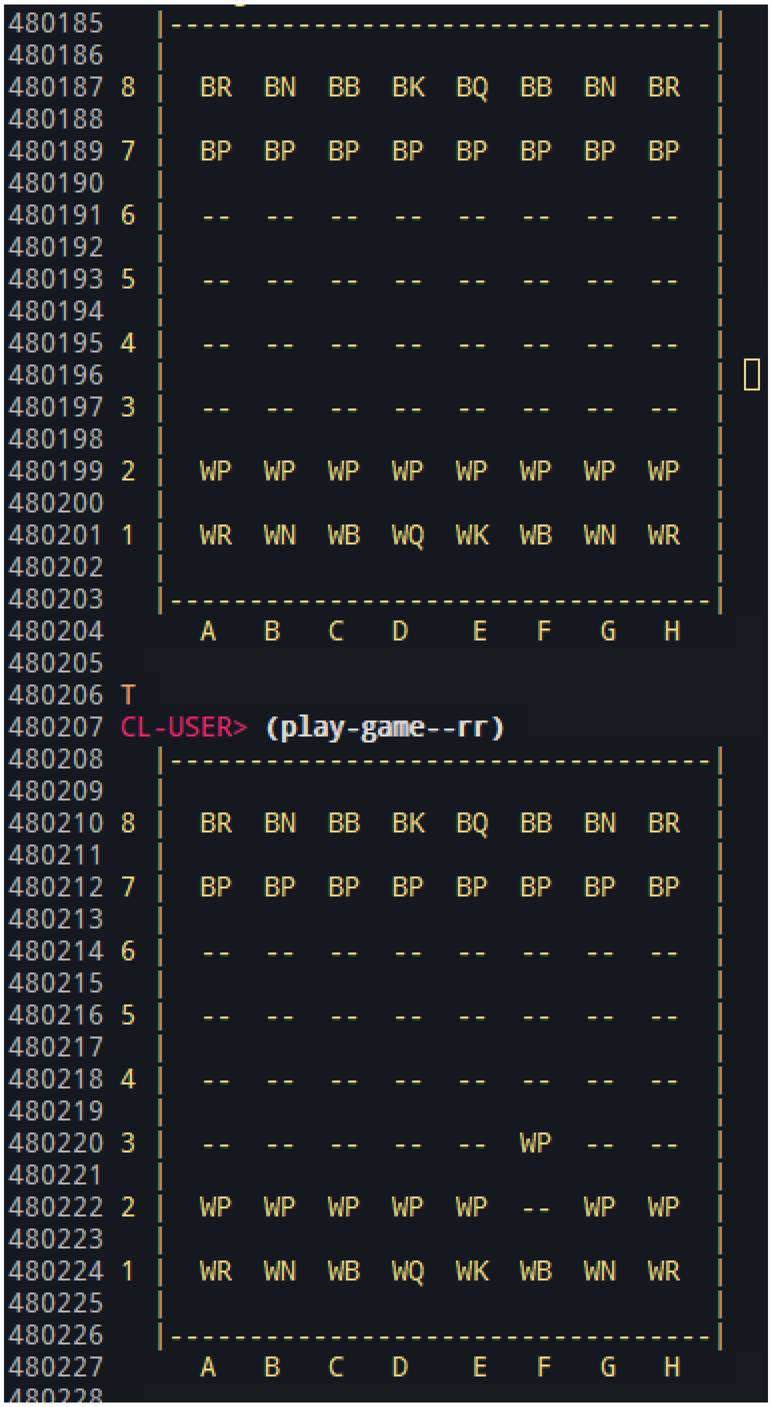
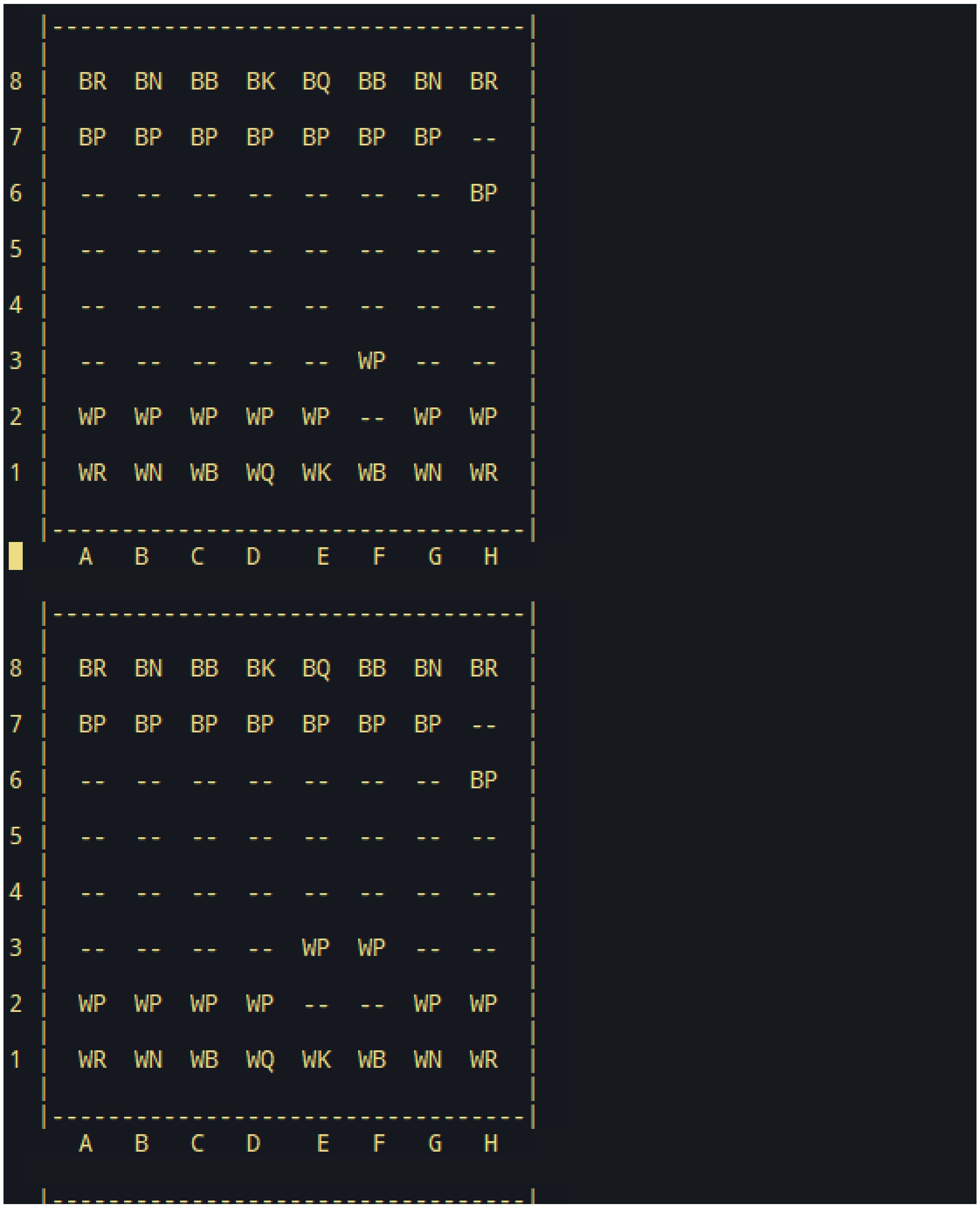
Chess AI Player Task 15: Create the Random Chess Player

Abstract: The goal of this task was to create the random player. This player creates a list of all of the possible moves and randomly selects one of them. The random player is capable of playing against a human or another random player.

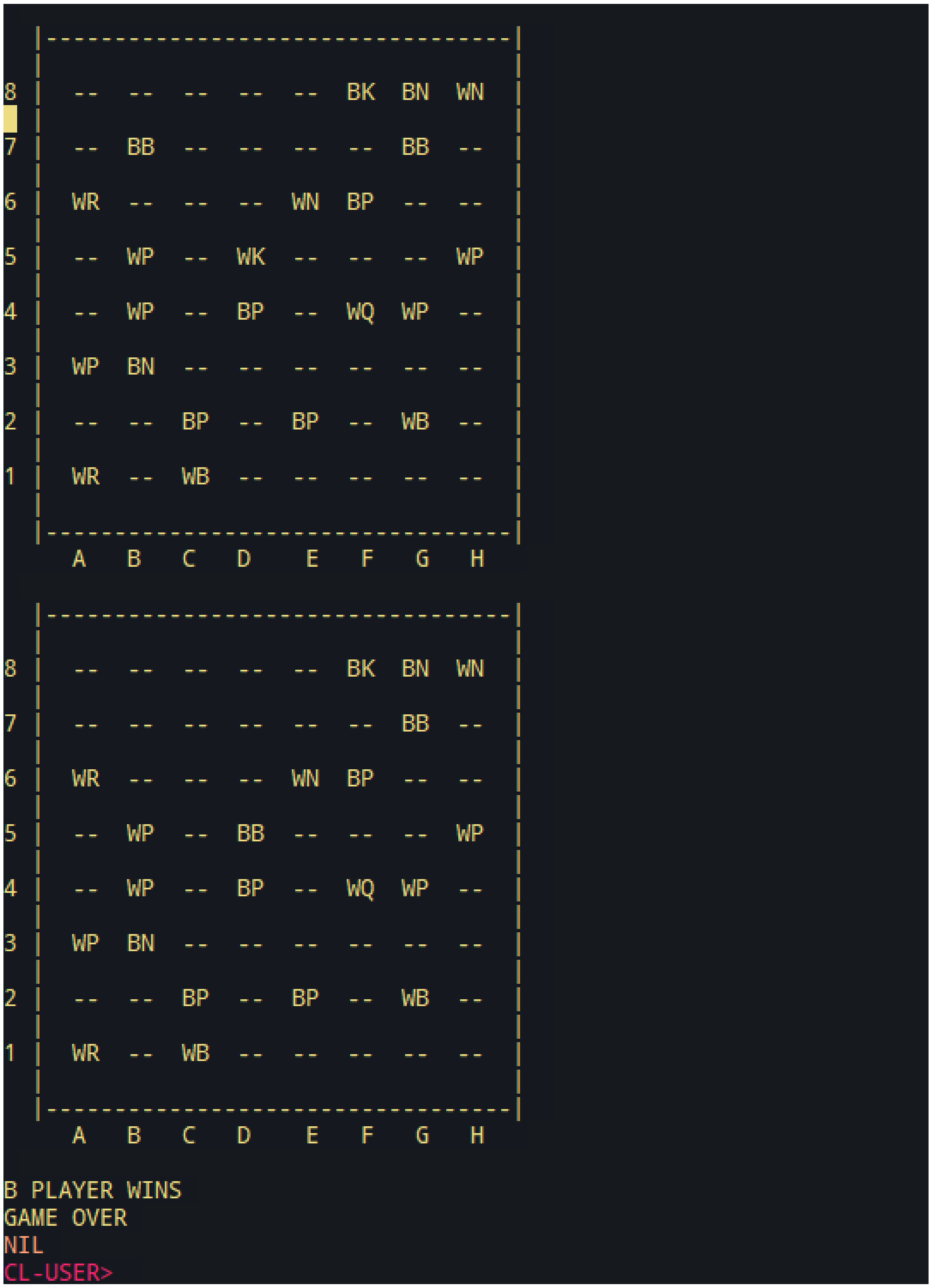
Demo:



```
   |--------------------------------|
   |                                |
 8 |  BR  BN  BB  BK  BQ  BB  BN  BR |
   |                                |
 7 |  BP  BP  BP  BP  BP  BP  BP  BP |
   |                                |
 6 |  --  --  --  --  --  --  --  -- |
   |                                |
 5 |  --  --  --  --  --  --  --  -- |
   |                                |
 4 |  --  --  --  --  --  --  --  -- |
   |                                |
 3 |  --  --  --  --  --  --  --  -- |
   |                                |
 2 |  WP  WP  WP  WP  WP  WP  WP  WP |
   |                                |
 1 |  WR  WN  WB  WQ  WK  WB  WN  WR |
   |                                |
   |--------------------------------|
      A   B   C   D    E   F   G   H

T
CL-USER> (play-game--rr)
   |--------------------------------|
   |                                |
 8 |  BR  BN  BB  BK  BQ  BB  BN  BR |
   |                                |
 7 |  BP  BP  BP  BP  BP  BP  BP  BP |
   |                                |
 6 |  --  --  --  --  --  --  --  -- |
   |                                |
 5 |  --  --  --  --  --  --  --  -- |
   |                                |
 4 |  --  --  --  --  --  --  --  -- |
   |                                |
 3 |  --  --  --  --  --  WP  --  -- |
   |                                |
 2 |  WP  WP  WP  WP  WP  --  WP  WP |
   |                                |
 1 |  WR  WN  WB  WQ  WK  WB  WN  WR |
   |                                |
   |--------------------------------|
      A   B   C   D    E   F   G   H
```



```
   |---------------------------------|
   |                                 |
8  |  BR  BN  BB  BK  BQ  BB  BN  BR |
   |                                 |
7  |  BP  BP  BP  BP  BP  BP  BP  -- |
   |                                 |
6  |  --  --  --  --  --  --  --  BP |
   |                                 |
5  |  --  --  --  --  --  --  --  -- |
   |                                 |
4  |  --  --  --  --  --  --  --  -- |
   |                                 |
3  |  --  --  --  --  --  WP  --  -- |
   |                                 |
2  |  WP  WP  WP  WP  WP  --  WP  WP |
   |                                 |
1  |  WR  WN  WB  WQ  WK  WB  WN  WR |
   |                                 |
   |---------------------------------|
      A   B   C   D    E   F   G   H

   |---------------------------------|
   |                                 |
8  |  BR  BN  BB  BK  BQ  BB  BN  BR |
   |                                 |
7  |  BP  BP  BP  BP  BP  BP  BP  -- |
   |                                 |
6  |  --  --  --  --  --  --  --  BP |
   |                                 |
5  |  --  --  --  --  --  --  --  -- |
   |                                 |
4  |  --  --  --  --  --  --  --  -- |
   |                                 |
3  |  --  --  --  --  WP  WP  --  -- |
   |                                 |
2  |  WP  WP  WP  WP  --  --  WP  WP |
   |                                 |
1  |  WR  WN  WB  WQ  WK  WB  WN  WR |
   |                                 |
   |---------------------------------|
      A   B   C   D    E   F   G   H

   |---------------------------------|
```



```
  |---------------------------------|
  |                                 |
8 |  --  --  --  --  --  BK  BN  WN |
  |                                 |
7 |  --  BB  --  --  --  --  BB  -- |
  |                                 |
6 |  WR  --  --  --  WN  BP  --  -- |
  |                                 |
5 |  --  WP  --  WK  --  --  --  WP |
  |                                 |
4 |  --  WP  --  BP  --  WQ  WP  -- |
  |                                 |
3 |  WP  BN  --  --  --  --  --  -- |
  |                                 |
2 |  --  --  BP  --  BP  --  WB  -- |
  |                                 |
1 |  WR  --  WB  --  --  --  --  -- |
  |                                 |
  |---------------------------------|
     A   B   C   D    E   F   G   H

  |---------------------------------|
  |                                 |
8 |  --  --  --  --  --  BK  BN  WN |
  |                                 |
7 |  --  --  --  --  --  --  BB  -- |
  |                                 |
6 |  WR  --  --  --  WN  BP  --  -- |
  |                                 |
5 |  --  WP  --  BB  --  --  --  WP |
  |                                 |
4 |  --  WP  --  BP  --  WQ  WP  -- |
  |                                 |
3 |  WP  BN  --  --  --  --  --  -- |
  |                                 |
2 |  --  --  BP  --  BP  --  WB  -- |
  |                                 |
1 |  WR  --  WB  --  --  --  --  -- |
  |                                 |
  |---------------------------------|
     A   B   C   D    E   F   G   H

B PLAYER WINS
GAME OVER
NIL
CL-USER>
```

Code:

```
( defun random-move ( move-pairs &aux curr-square dest
square selected )
  ( setf selected ( nth ( random ( length move-pairs ) )
move-pairs ) )
  ( setf curr-square ( car selected ) )
  ( setf dest-square ( car ( cdr selected ) ) )
  ( move curr-square dest-square )
)

( defmethod get-move-pair-list ( ( piece piece ) &aux
curr-square poss-dests )
  ( setf curr-square ( cs piece ) )
  ( setf poss-dests ( possible-moves piece ) )
  ( mapcar ( lambda ( dest ) ( list curr-square dest ) )
poss-dests )
)


( defun play-turn--rr ()
  ( if ( game-overp )
    ( progn
       ( format t "GAME OVER" )
       nil
     )
     ( progn
       ( random-white-move )
       ( if ( game-overp )
         ( progn
           ( format t "GAME OVER" )
           nil
          )
          ( progn
           ( random-black-move )
           ( play-turn--rr )
          )
       )
     )
  )
)

( defun play-turn--hr ( color &aux curr-square csr csf
dest-square dsr dsf )
  ( if ( game-overp )
     ( progn
       ( format t "GAME OVER" )
       nil
     )
     ( progn
       ( format t "It is the ~A player's turn~%" color )
       ( format t "Enter start square: " )
       ( setf curr-square ( parse-square ( string-trim " "
(read-line))))
       ( setf csr ( car curr-square ) )
       ( setf csf ( car ( cdr curr-square ) ) )
       ( format t "Enter end square: " )
       ( setf dest-square (parse-square (string-trim " "
(read-line))))
       ( setf dsr ( car dest-square ) )
       ( setf dsf ( car ( cdr dest-square ) ) )
       ( setf curr-square ( aref ( board *gameboard* ) csf
csr ) )
       ( setf dest-square ( aref ( board *gameboard* ) dsf
dsr ) )
       ( move curr-square dest-square )
       ( random-black-move )
       ( play-turn--hr 'w )
     )
   )
)


( defun play-game--hr ()
  ( play-turn--hr 'w )
)

( defun play-game--rr ()
  ( play-turn--rr )
)


( defun random-black-piece ()
  ( nth ( random ( length *black-pieces* ) ) *black-pieces*
)
)

( defun random-black-move ()
  ( random-move ( get-all-color-moves 'b ) )
)

( defun random-white-move ()
  ( random-move ( get-all-color-moves 'w ) )
)

( defun pieces-of-color ( color )
  ( cond
    ( ( eq color 'w ) *white-pieces* )
     ( ( eq color 'b ) *black-pieces* )
  )
)

( defun oppo-pieces-of-color ( color )
  ( cond
    ( ( eq color 'b ) *white-pieces* )
     ( ( eq color 'w ) *black-pieces* )
  )
)

(defun get-oppo-color-moves (color)
  (cond
    ( ( eq color 'w ) ( get-all-color-moves 'b ) )
     ( ( eq color 'b ) ( get-all-color-moves 'w ) )
  )
)

( defun oppo-color ( color )
  ( cond
    ( ( eq color 'w ) 'b )
     ( ( eq color 'b ) 'w )
  )
)
```