Project Task 11:
Minimax Part 1
By Carrie Corcoran

This is the seventh task for my semester- long project of creating mancala playing machines, and the first task related to Minimax. In this task, I defined how to save a game state and the static evaluation function. This task is relatively light on code as it came with additional research into Minimax AI.

Code:

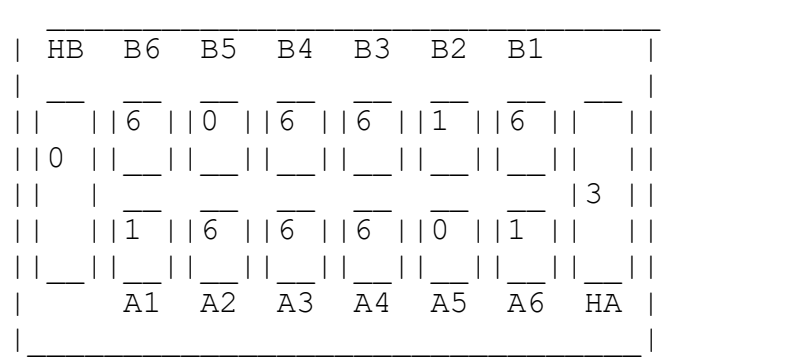
```
(defun save-state (current)
  (cond
    (current
       (setf state (list
(a1)(a2)(a3)(a4)(a5)(a6)(ah)(b1)(b2)(b3)(b4)(b5)(b6)(bh)))
     )
     (t
       (setf state (list (get a1-copy value)(get a2-copy
value)(get a3-copy value)(get a4-copy value)
       (get a5-copy value)(get a6-copy value)(get ah-copy
value)(get b1-copy value)
       (get b2-copy value)(get b3-copy value)(get b4-copy
value)(get b5-copy value)
       (get b6-copy value)(get bh-copy value)))
     )
  )
  state
)

(defun static-eval-function (state player)
  (cond
    ((eql player 'a)
       (-(nth 6 state) (nth 13 state))
     )
     (t
       (-(nth 13 state) (nth 6 state))
     )
  )
)

(defun create-node (player state move)
  (list player state move (static-eval-function state player))
)
```

Demo:

```
[]> (static-eval-test)
Testing starting state for player A: 0
Making move:
```




```
Testing state after move for player A: 3
Testing state after move for player B: -3
NIL
```