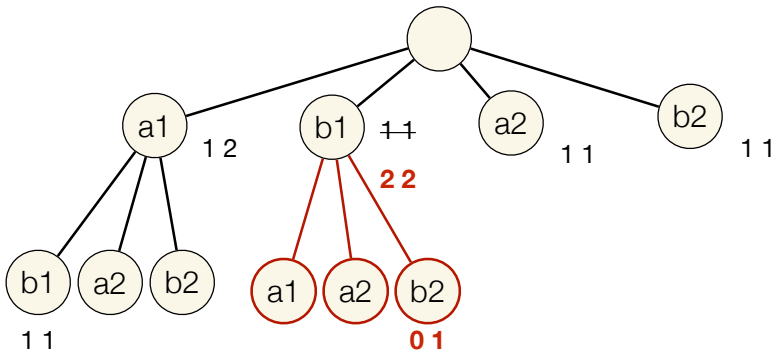
1. The following code is from `hex/hex_vc3.py` in the examples github:

```
1  while len(mustplay) > 0:
2    for move in CELLS:
3      if move in mustplay: break
```

What is the purpose of lines 2 and 3?
They ensure that the cells in `mustplay` are iterated over in the order defined by `CELLS`. (Recall that some orders are much more efficient for solving Hex than others).

2. Below is the Monte Carlo search tree after the first 5 simulations of running the `g x` command of `mcts/main.py` with a 2×2 board. Show the tree after simulation 6 (output below); explain each change. The indices 5,6,9,10 correspond to positions `a1`, `b1`, `a2`, `b2` respectively.




```
trv_xpnd bu * .8 1.4 1.4 1.4 6
  xpnd_nd * 6 > 5
  xpnd_nd * 6 > 9
  xpnd_nd * 6 > 10
sim 6. * 6 10 roll 9 parent loss
```

New nodes and values are in red. The `b1` node in the search tree was *expanded* by adding the children `a1`,`a2`,`b2`. The new child `b2` was selected for rollout; the simulation resulted in a loss for the parent of `b2`, so we add 0 wins and 1 visits to `b2`'s counter. The results are back-propagated to the parent by adding 1 to both wins and visits (since a loss for `b2`'s parent is a win for their grandparent).

3. In the example code `mcts/mcts1.py`:

   (a) What is the difference between `get_best_move` and `best_uct`?
   The `best_uct` method uses UCT (i.e., winrate + exploration bonus) to select a child within the search tree to traverse; it is called repeatedly until a leaf in the search tree is reached. The `get_best_move` method is used after search to select a single move from the root node to actually play; it returns the child of the root with the most visits.

   (b) Consider the following code from the `best_uct` method:

```
# calculate UCT, update if best
mean_res = child.results / child.sims
uct = ___(1)___ + (self.c * sqrt(log(self.root_node.sims) / child.sims))
if best_uct is None or uct > best_uct:
    best_uct, best_child = uct, child
```

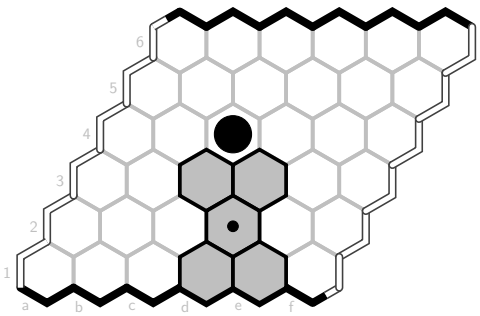
   What should missing expression (1) be? `mean_res`

   (c) `self.c` gets initialized to 0.3. What would be the effect of setting it to something much larger (e.g., 50)?
   Making `c` larger makes the exploration bonus term larger. This means that exploration will be weighted more heavily than exploitation; relatively less-visited nodes will be chosen more often even if their winrates are low than they would have been with a smaller value of `c`.

(d) If you run `mcts/main.py` multiple times and ask for a winning move for Black (using the command `g x`), it will sometimes generate different moves in different runs. Why?

MCTS chooses nodes based on their estimated quality. However, their quality is estimated based on random rollouts, which are noisy; on different runs, any given random rollout may give a different result, which can cause nodes that appear high-quality in some runs to appear low-quality in others. (This effect is strongest at the beginning, when every node has relatively few rollouts. The more rollouts are performed, the more accurate and less noisy the quality estimates will become.)

4. Consider the following virtual connection in 6×6 Hex:



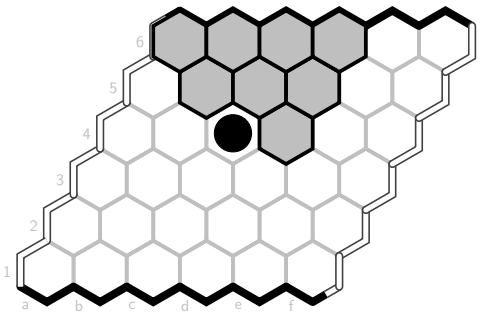
(a) Does this depict a *full connection* or a *semi-connection* between the Black stone and the bottom border? Explain your reasoning.

This is a *semi-connection*: If it is currently Black's turn, they can play on d2, which is strongly connected to c4 by a bridge pattern (c3 and d3) and also to the bottom border by another bridge pattern (d1 and e1). However, if it is White's turn currently, they can block this strategy by playing on d2.

(b) Write this connection in logical form as an AND/OR strategy.

$d2 \wedge (c3 \vee d3) \wedge (d1 \vee e1)$

5. Consider the following virtual connection in 6×6 Hex:



(a) Does this depict a *full connection* or a *semi-connection* between the Black stone and the bottom border? Explain your reasoning.

This is a *full connection* from c4 to the top border, via a 2-3-4 pattern.

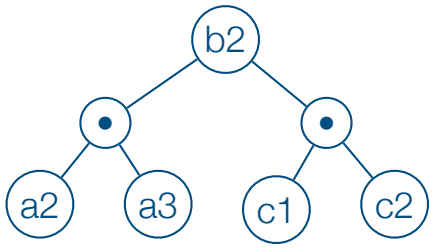
(b) Write this connection in logical form as an AND/OR strategy.

$\big(b5 \wedge (a6 \vee b6)\big) \vee \big(d5 \wedge (c5 \vee d4) \wedge (c6 \vee d6)\big)$

6. Consider the following AND/OR strategy for the 3×3 Hex position after `1.B[a1]`.

$$b2 \wedge (a2 \vee a3) \wedge (c1 \vee c2)$$

(a) Draw the strategy as an AND/OR tree.



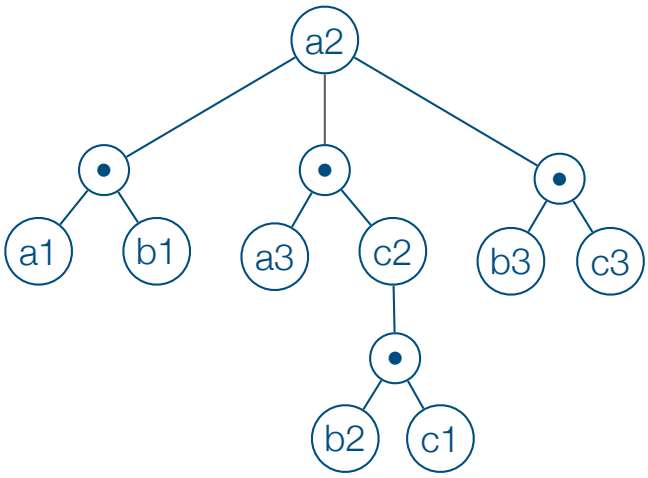
(b) Is this is a winning strategy for Black, White, or neither?

This is a winning strategy for White: every cellset connects the `left` border to the `right` border, and no substrategies of an OR-expression have overlapping cellsets.

7. Consider the following AND/OR strategy for 3×3 Hex:

$$a2 \wedge (b3 \vee c3) \wedge (a1 \vee b1) \wedge (a3 \vee (c2 \wedge (b2 \vee c1)))$$

(a) Draw the strategy as an AND/OR tree.



(b) How should Black reply to `1.B[a2] 2.W[c1]`?

a3: The OR-substrategy O_j whose carrier contains `c1` is $(a3 \vee (c2 \wedge (b2 \vee c1)))$. The AND-substrategy A_x of O_j that does *not* contain `c1` is `a3`.

(c) How should Black reply to `1.B[a2] 2.W[a1]`?

b1: The OR-substrategy O_j whose carrier contains `a1` is $(a1 \vee b1)$. The AND-substrategy A_x of O_j that does not contain `a1` is `b1`.

8. Recall that in Rock Paper Scissors, the winner gets 1 point, the loser gets -1 points, and both players get 0 in a draw. In the table below, the first utility in each cell is for the Row player, and the second utility is for the Column player.

	R	P	S
R	$0, 0$	$-1, 1$	$1, -1$
P	$1, -1$	$0, 0$	$-1, 1$
S	$-1, 1$	$1, -1$	$0, 0$

Rock Paper Scissors

(a) Is R an optimal strategy for the Row player? Why or why not?

No. If Column best-responds to R (with P), then Row will get -1. But if Row plays each of R, P, and S with $1/3$ probability, then Row's expected utility will be 0 no matter what Column plays.

(b) Suppose that the Column player is playing a mixed (randomized) strategy in which they play R with $1/3$ probability, P with $1/3$ probability, and S with $1/3$ probability. What is Row's expected utility for playing R against this strategy?

If Column plays strategy t which randomizes as described and Row plays R, then Row's expected utility will be

$$\begin{aligned} u_r(R, t) &= \frac{1}{3}u_r(R, R) + \frac{1}{3}u_r(R, P) + \frac{1}{3}u_r(R, S) \\ &= \frac{1}{3}0 + \frac{1}{3}(-1) + \frac{1}{3}1 \\ &= 0 \end{aligned}$$

(c) Suppose that the Column player is playing a mixed (randomized) strategy in which they play R with .5 probability, P with .3 probability, and S with .2 probability. What is Row's expected utility for playing R against this strategy?

If Column plays strategy t which randomizes as described and Row plays R, then Row's expected utility will be

$$\begin{aligned} u_r(R, t) &= 0.5u_r(R, R) + 0.3u_r(R, P) + 0.2u_r(R, S) \\ &= 0.5(0) + 0.3(-1) + 0.2(1) \\ &= -0.1 \end{aligned}$$