1. This question considers a variant of Tic-Tac-Toe, called Notakto, in which both players alternately place a cross (no one uses circles) on a 3-by-3 board, and the player that produces 3 crosses in a row (horizontal, vertical or diagonal) *loses* the game.

   After 3 moves, the following game state s_0 is given, with player $p(s_0) = -1$ making the next move:

	X	X
	X	

   [You may want to draw the decision tree starting at this state. Considering transpositions and symmetries can make this much easier!]

   (a) (3 points) What is the limit of the random rollout value $\lim_{N(s_0)\to\infty} \tilde{V}_{\mathsf{R}}(s_0)$?

   Solution: Let $V'(s) := \lim_{N(s)\to\infty} \tilde{V}_{\mathsf{R}}(s)$ and let $[x, y] \in \mathcal{A}$ denote the action of making a cross at column $x \in \{1, 2, 3\}$ and row $y \in \{1, 2, 3\}$. It is useful to draw a decision tree and to mark the actions that would end the game with a dot (or something). Let furthermore $s_1 = \mathcal{P}(s_0, [1, 3])$, $s_2 = \mathcal{P}(s_0, [3, 1])$, $s_3 = \mathcal{P}(s_0, [3, 3])$ and $s_4 = \mathcal{P}(s_0, [1, 1])$. Note that $s_2 = \phi(s_1)$ is (diagonal mirror) symmetric to s_1 and we only have to follow one. The other reachable states are $s_5 = \mathcal{P}(s_2, [1, 1]) = \mathcal{P}(s_4, [1, 3])$ and $s_6 = \mathcal{P}(s_4, [3, 1]) = \phi(s_5)$, which is a (diagonal mirror) symmetric to s_5. So there are only two (unique) states where the next action will always end the game: $V'(s_5) = +1$ and $V'(s_3) = -1$.

s_1

	.	X
.	X	X
.	X	.

s_2

	.	.
.	X	X
X	X	.

s_3

.	.	.
.	X	X
.	X	X

s_4

X	.	
.	X	X
	X	.

s_5

X	.	.
.	X	X
X	X	.

s_6

X	.	X
.	X	X
.	X	.

$$
\begin{aligned}
V'(s_0) &= \tfrac{2}{6} + \tfrac{2}{6}V'(s_2) + \tfrac{1}{6}V'(s_3) + \tfrac{1}{6}V'(s_4) \\
&= \tfrac{2}{6} + \tfrac{2}{6}\big(-\tfrac{4}{5} + \tfrac{1}{5}V'(s_5)\big) + \tfrac{1}{6}V'(s_3) + \tfrac{1}{6}\big(-\tfrac{3}{5} + \tfrac{2}{5}V'(s_5)\big) \\
&= \tfrac{2}{6} - \tfrac{8}{30} + \tfrac{2}{30} - \tfrac{1}{6} - \tfrac{3}{30} + \tfrac{2}{30} \quad = \quad \tfrac{10-8+2-5-3+2}{30} \quad = \quad -\tfrac{1}{15}
\end{aligned}
$$

   (b) (2 points) Next we implement a smart rollout heuristic: whenever possible, we select actions that do not end the game. What is the limit of the random rollout value using this heuristic?

   Solution: Using the nomenclature above, the value in question is now

$$
\begin{aligned}
V'(s_0) &= \tfrac{2}{4}V'(s_2) + \tfrac{1}{4}V'(s_3) + \tfrac{1}{4}V'(s_4) \\
&= \tfrac{2}{4}V'(s_5) + \tfrac{1}{4}V'(s_3) + \tfrac{1}{4}\big(\tfrac{1}{2}V'(s_5) + \tfrac{1}{2}V'(s_6)\big) \\
&= \tfrac{2}{4} - \tfrac{1}{4} + \tfrac{1}{4} \quad = \quad \tfrac{1}{2}
\end{aligned}
$$

   (c) (2 points) What is the minmax value $V(s_0)$? What is the effect of the above heuristic rollout on the minmax value? Explain your answer.

Solution: As player -1 can choose the next move, he/she can go to s_3, where he/she is guaranteed to win. The minmax value is therefore $V(s_0) = -1$. The rollout heuristic does not affect the minmax value of a state in any way.

(d) (3 points) Override the following standard implementation of `rollout()` to implement the smart rollouts of the above sub-question (b). Make sure you implement the above heuristic, and do not use an abstract heuristic $H(s, a)$.

```
def rollout(self, node):
    state = node.state
    while not state.terminal():
        action = random.choice(state.actions())
        state = state.transition(action)
    return state.reward()
```

Solution:

```
def smart_action(self, state):
  allow, lose = [], []
  for a in state.actions():
    if state.transition(a).reward() < 0:
      lose.append(a)
    else:
      allow.append(a)
  return random.choice(allow if len(allow) > 0 else lose)

def rollout(self, node):
    state = node.state
    while not state.terminal():
        state = state.transition(self.smart_action(state))
    return state.reward()
```