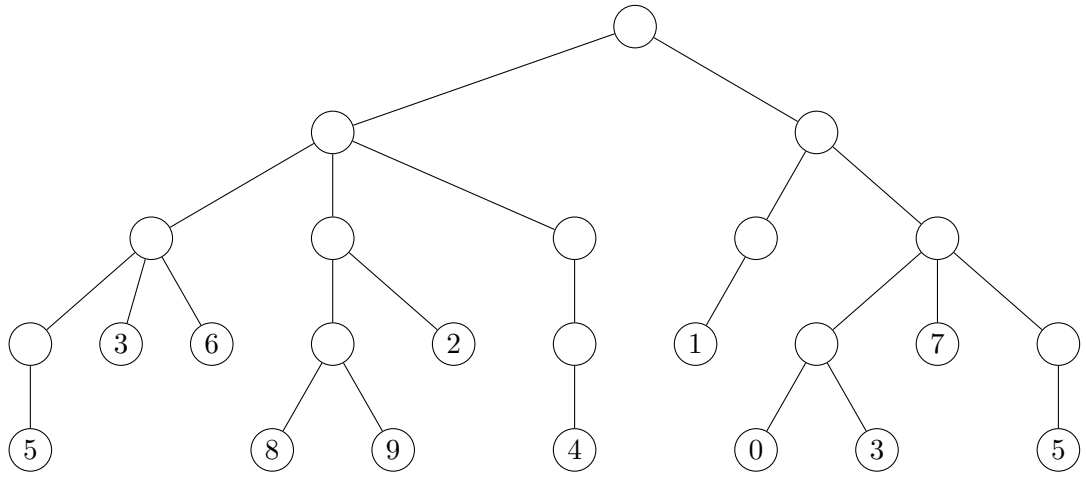
1. **Label** each node with its minimax (not negamax) value. The leafs are done for you. Root is a max node.



The minimax value of a game state is **(circle one)**

a) the minimum score the player can achieve, over all possible opponent strategies

b) the maximum score the player can achieve, over all possible opponent strategies

c) the minimum score the player can achieve against a typical maximizing opponent

d) the maximum score the player can achieve against a typical minimizing opponent

Minimax is often implemented with enhancements, because **(circle one)**

a) bottom-up algorithms tend to be inefficient

b) the minimax definition does not consider opponent modelling

c) the value can often be found without examining the whole search space

d) the search space is often too small for minimax to handle properly

2. Draw an unreachable tic-tac-toe state here:

It is unreachable because _____ .

In tic-tac-toe there are 9 possible 1st moves, each with 8 possible 2nd moves, and so on, so the number of different games

is at most **(give number or expression)** _____ .

When solving tic-tac-toe, using a _____ avoids

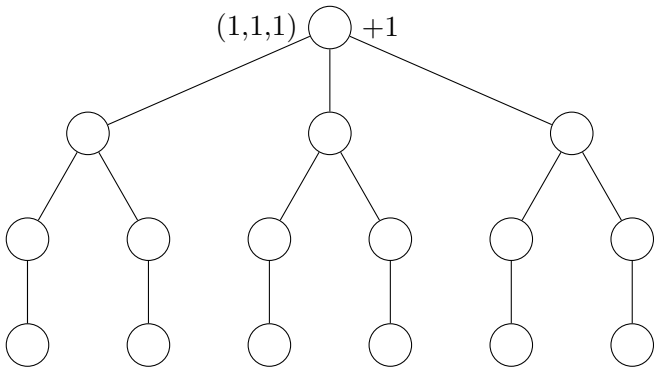
unneeded recomputation: an efficient empty-board tic-tac-toe minimax explores only about **(circle one)**

30000 10000 3000 1000 500 different positions.

3. The White Doctor opening in the game of _____

was solved in 2007 by a group of scientists at the University of _____ .

```
def foo(a,b,c):
  if a==0 and b==0 and c==0: return -1
  sofar = -1
  for aa in range(0,a):
    sofar = max(sofar, -foo(aa,b,c))
    #if sofar==1: return 1
  for bb in range(0,b):
    sofar = max(sofar, -foo(a,bb,c))
    #if sofar==1: return 1
  for cc in range(0,c):
    sofar = max(sofar, -foo(a,b,cc))
    #if sofar==1: return 1
  return sofar
```



On the recursion tree for the call `foo(1,1,1)` above, **label** each node with parameters **to the left** and return value **to the right**. The root is done for you.

4. The function `foo` in the previous question uses algorithm **(circle one)**

minimax alphabeta negamax negamax-alphabeta

to return the minimax value for an arbitrary **(circle one)**

3-pile nim position tic-tac-toe position 3x3 sliding tile position 3x3 Go position .

The number of nodes in the recursion tree for `foo(3,3,3)` is closest to **(circle one)**

12 24 120 240 1200 2400 12000 24000 .

If `foo` is changed by uncommenting the three comments, then the number of nodes in the recursion tree for `foo(3,3,3)` is closest to **(circle one)**

24000 12000 2400 1200 240 120 24 12 .