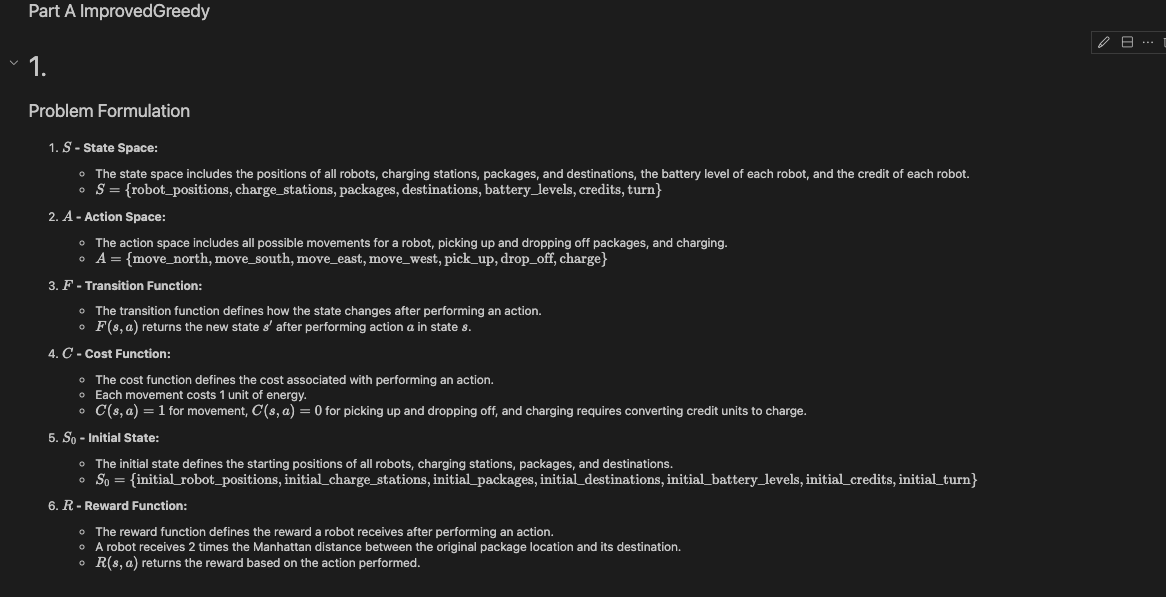
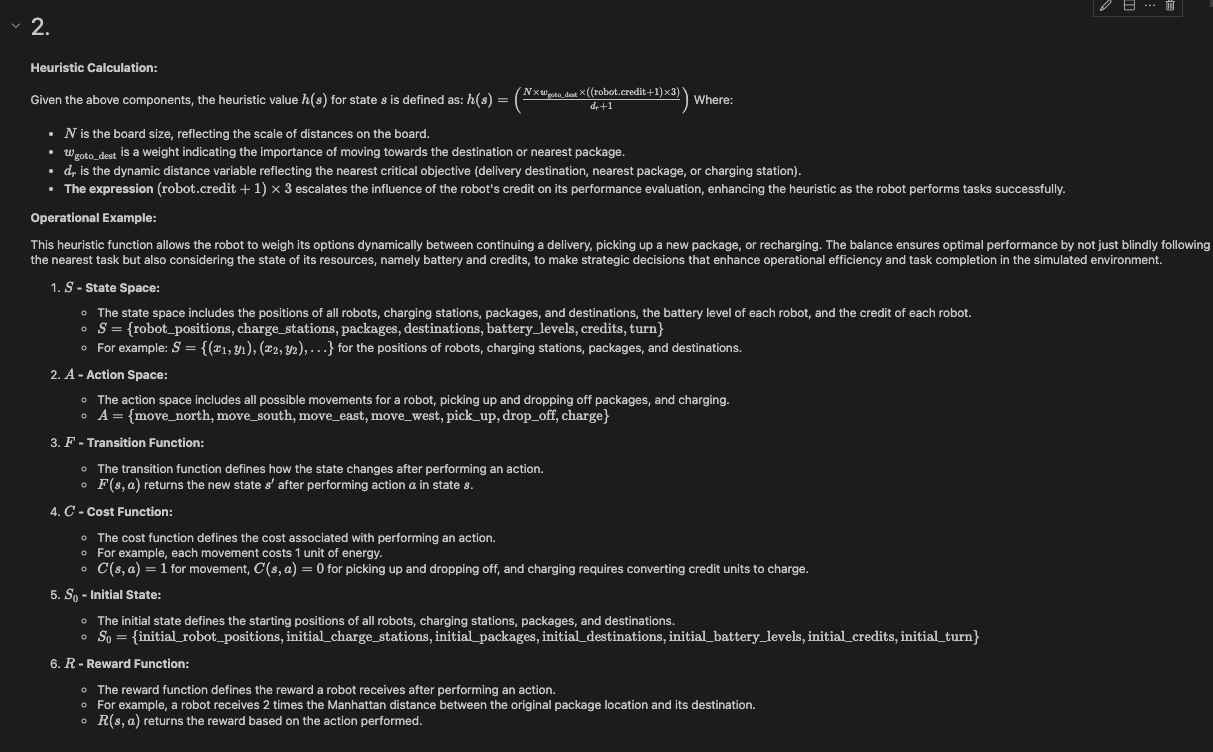
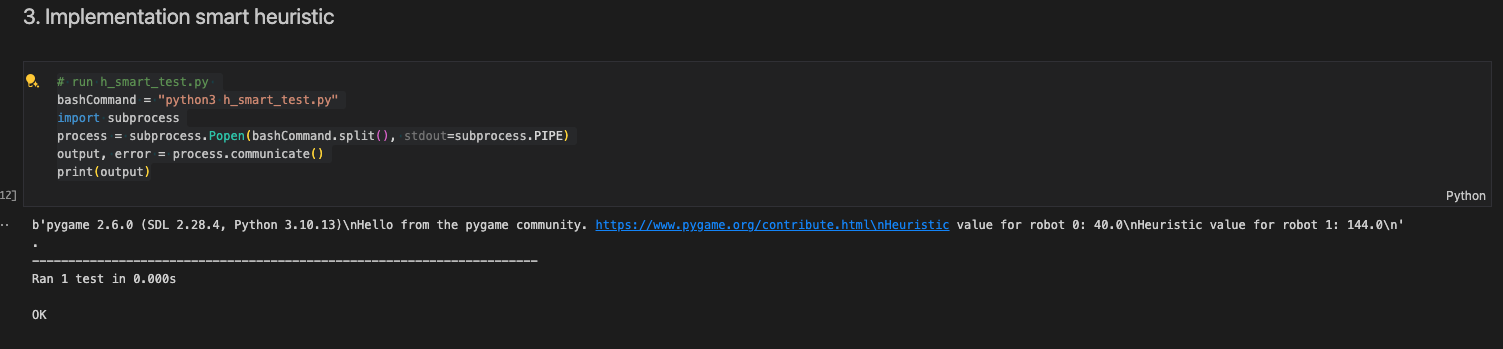
Intro AI

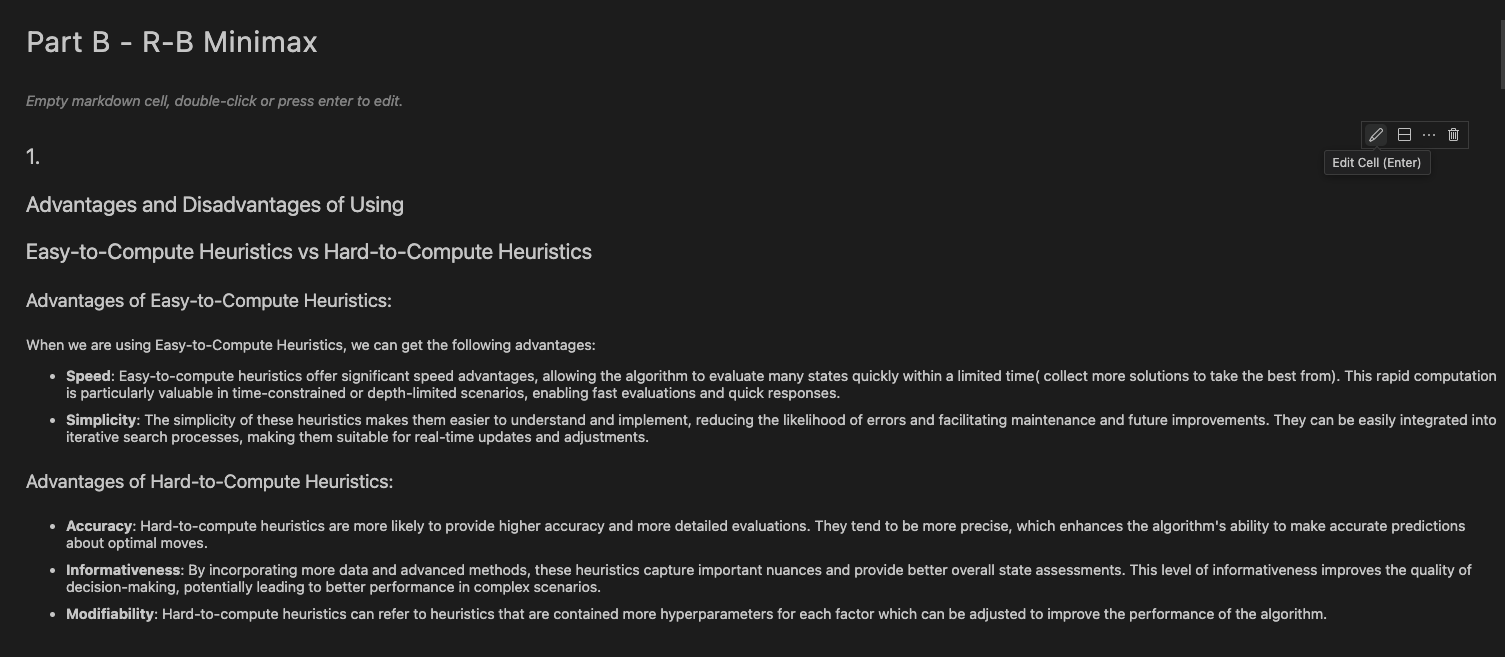
Amit levi

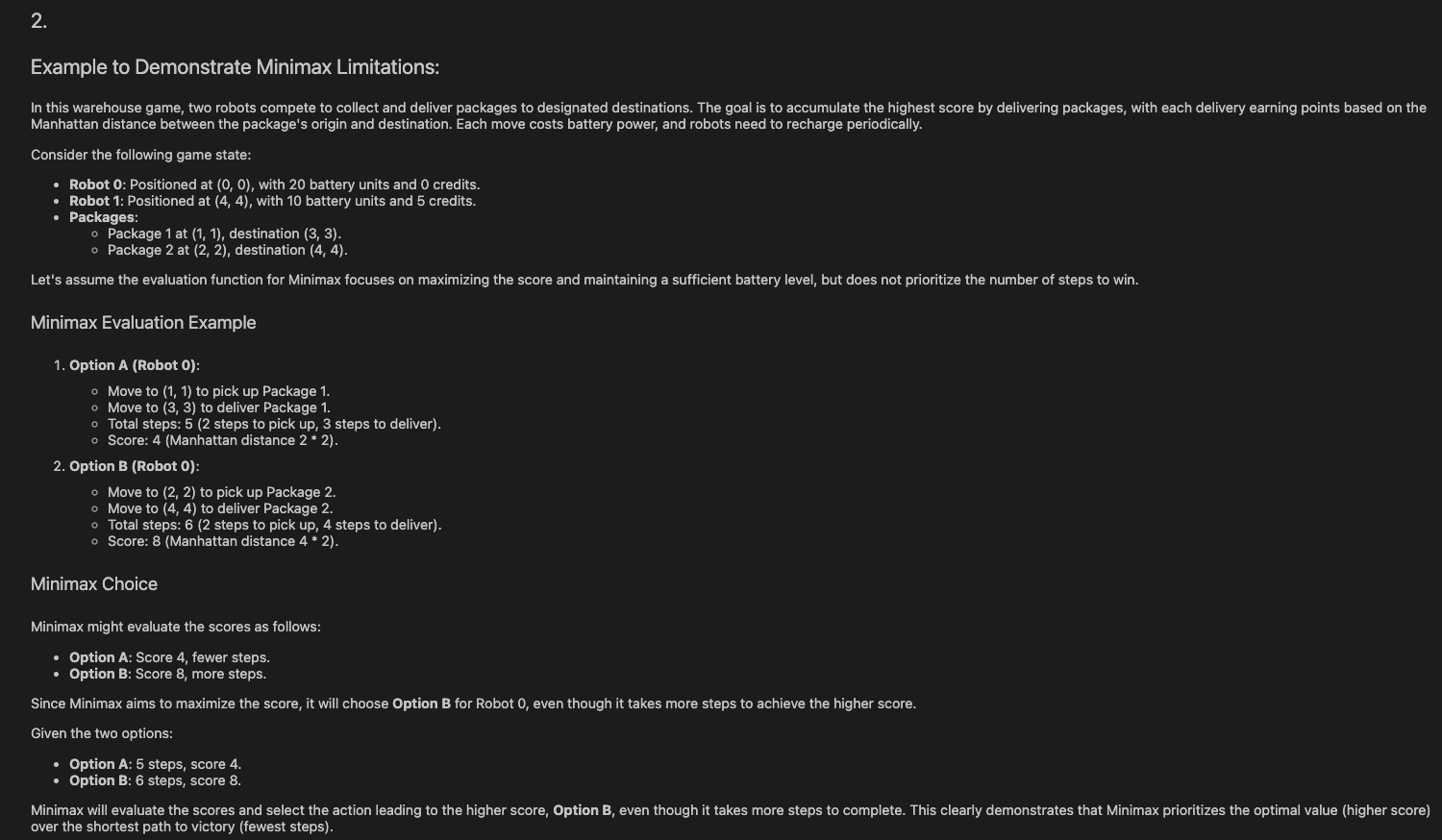
שחר כהן

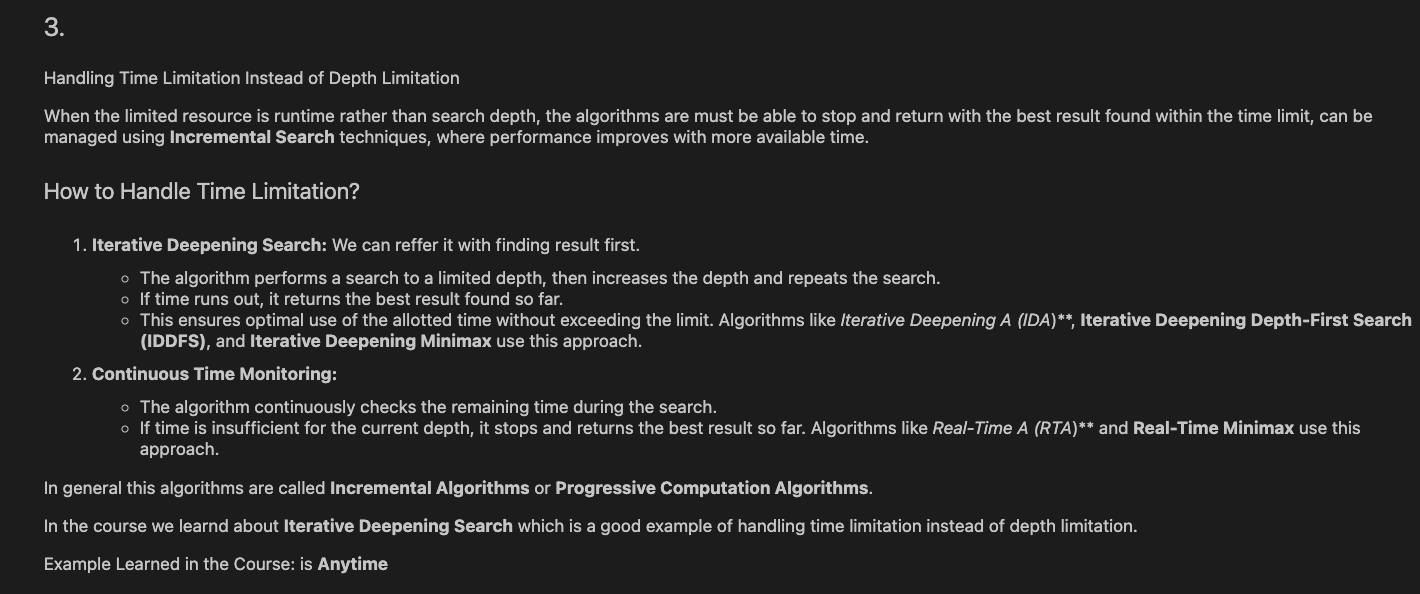


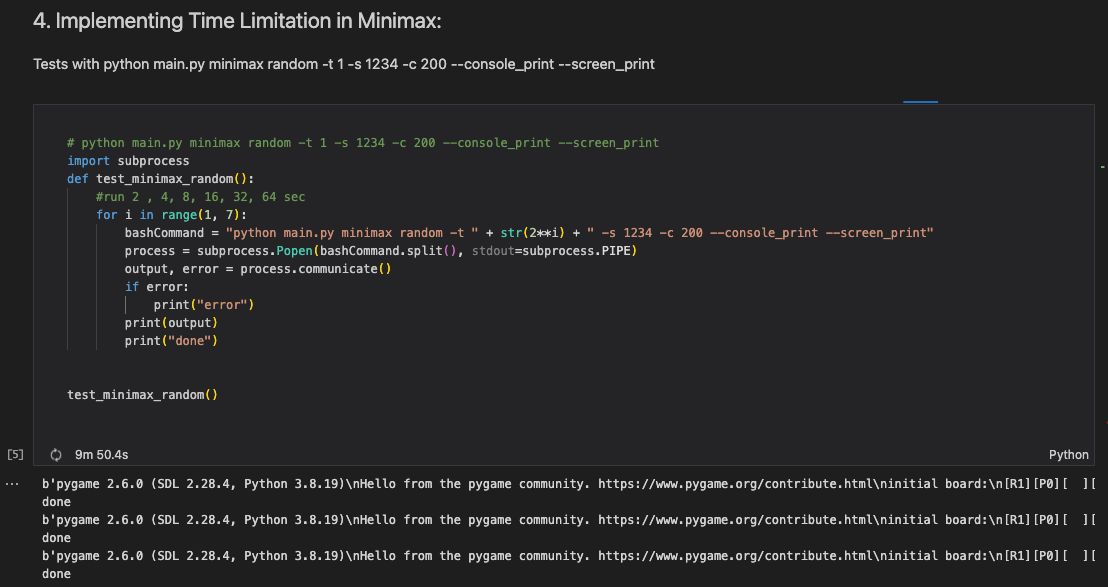


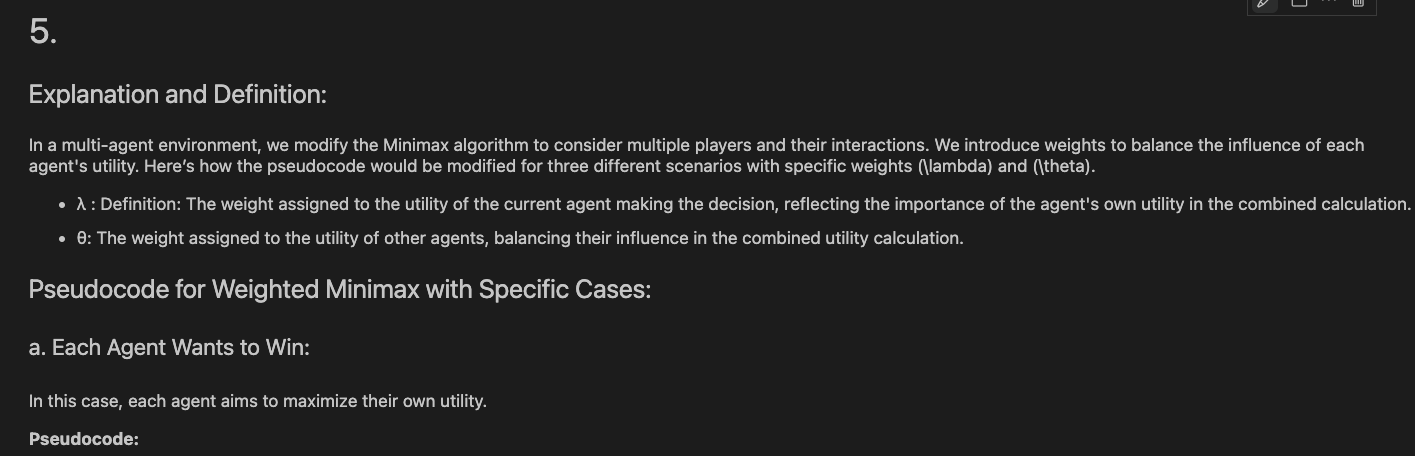


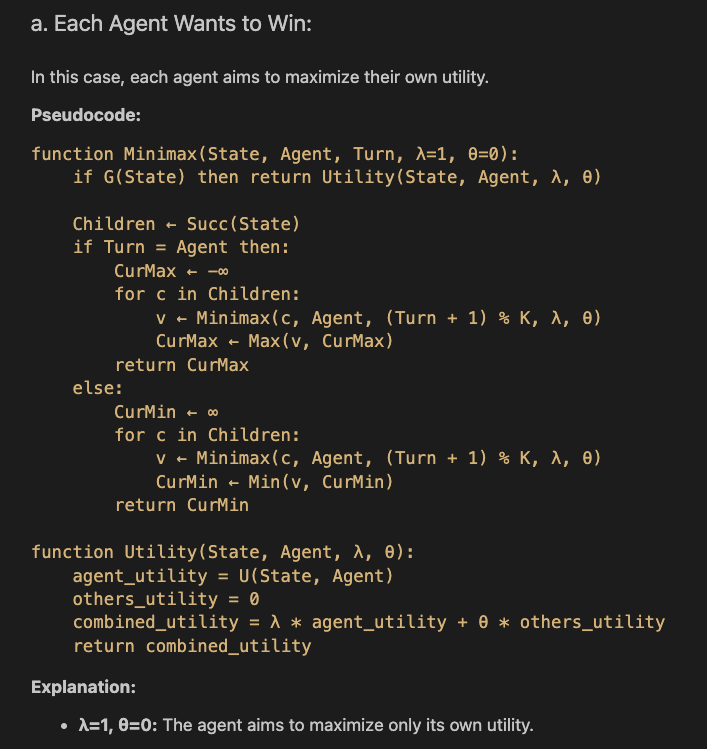


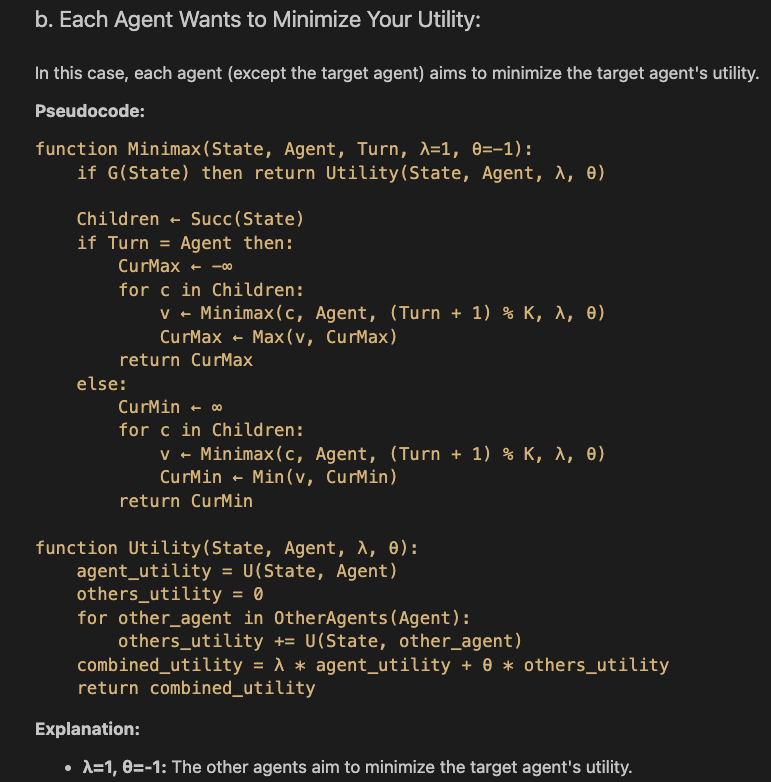


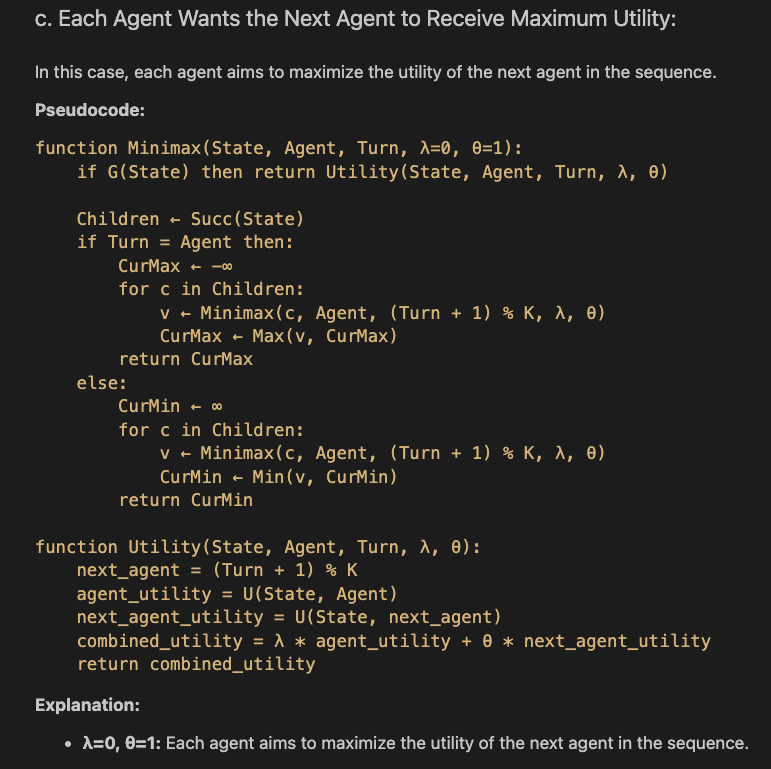


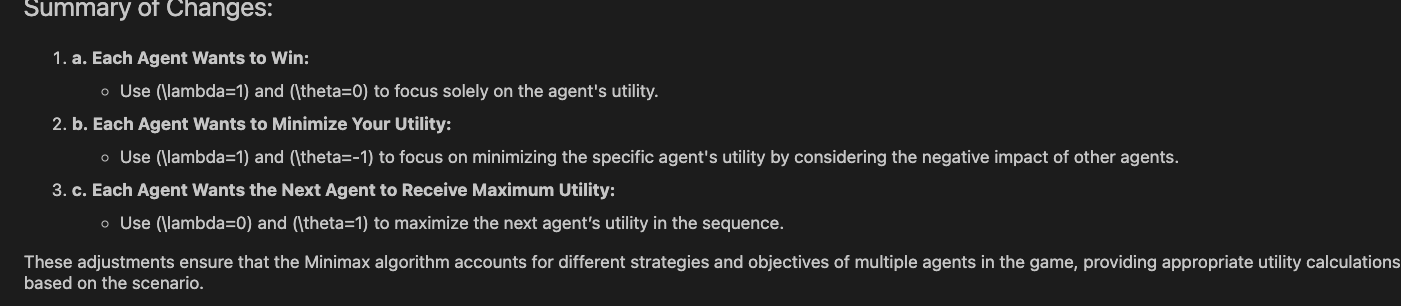


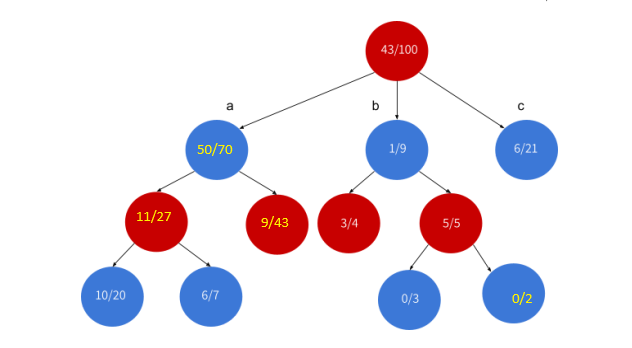


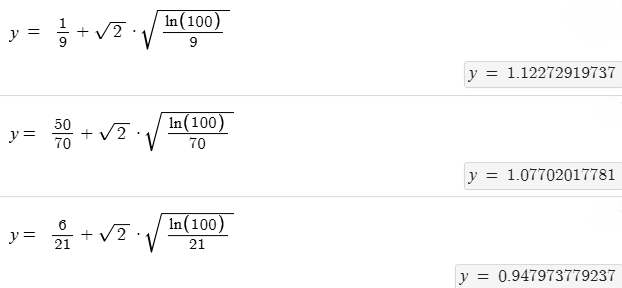








Question 6:  
  
1) 

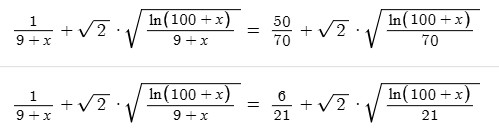
2) 

B

C

A

And so the select will expand on node B

3)   
 so the next node to be selected will be node A and I will happen after 1 simulation that blue  
 wins in.

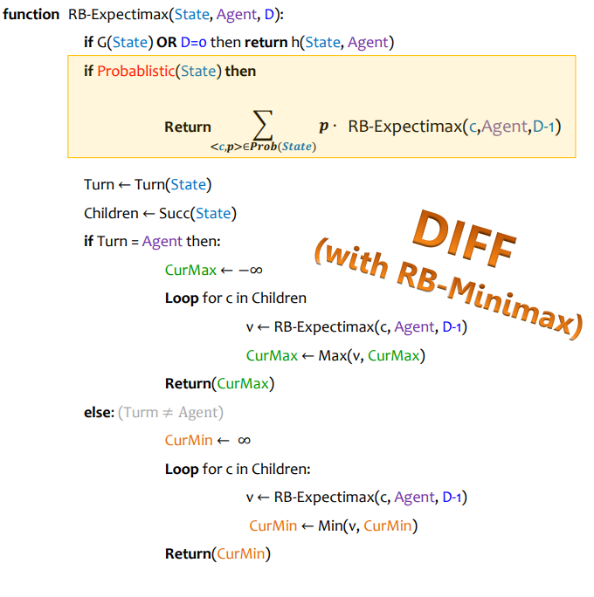
X = 3.285

X = 0.7161

Question 5:

1. A. The branching factor is defined to be the number of succ of a node v – when we look at the complexity we use its upper bound and thus we will refer to the upper bound of the branching factor which does not change in this case.  
   B. the Added operation of blocking adds another option to be explored and thus the branching factor increases by 1.
2. A. the use of AlphaBeta will be able to handle the added branching factor as it trims away branches anyways resulting in minimization of the branching factor impact on the overall time effect of the branching factor to some degree.   
   B. improved AlphaBeta in the following ways: anytime (as our game is time limited), child ordering by the previous information we have from anytime (to reduce the number of branches in need to check), closing libraries for branches with x blocking or more becase probablisticaly the branching factor can be very small for those cases as many squares will be blocked. These are the major improvements that can be implemented inorder to keep the time similar to that we had before the change.

Question 4:

1. In the case we are battling a randomly choosing operation opponent we will use a uniform distribution of probability in the expectimax algorithm for each action because the opponent chooses his at that probability and thus we effectively “weight” the utility of each node by its respective probability of happening.
2. Because the heuristic value is bounded the expectancy of the bots actions is well-defined and thus we can prune the path using the same method of alphabeta . We will use the calculated expectancy of the following node as the pruning factor values instead of the utility score. this will work because expectimax chooses based on the “known expectancy” of each probable node and thus we know that if we cut down on the those with low/ high expected values we will not affect the outcome result as we will not affect the slecetion process of expectimax. As can be seen bellow this is how we can compute such pruning:

Question 3:

Lets split the answers into cases: in the case that the bounded depth is reached by minmax then it will also be reached by alphabeta and the result will be the same. In the case that both of them fail to complete the same intended depth search then the answer returned will be that of the last depth and as before they will both return the same result. Because alphabeta prunes the tree and thus runs faster than minmax there is a chance that it will reach a depth bigger than minmax and thus its results will differ from that of minmax.

