HAROKOPIO UNIVERSITY
DEPARTMENT OF INFORMATION & TELEMATICS

Artificial Intelligence

2nd Task

Manousos Linardakis, it22064

Question 1:

pacman follows the minimax strategy in which he believes that the opponent is playing optimally. Sometimes pacman can "lose" with minimax because he decided that way he will get the highest score (if he is trapped by "optimal" ghosts or pressed for time). Also, pacman waits for the ghosts to approach him in minimax, since when they follow him (behind him), pacman can keep playing and not get trapped by them.

Question 2:

Since pacman is in a situation where he cannot win extra points (he is "trapped" by his opponents and thinks they are playing optimally) he decides that the best move is to "lose" as soon as possible, so that he does not lose and other points due to time. Minimax strategy is wrong when opponents are not playing optimally as it can lead to situations where pacman wins more points.

For example, if pacman considers the opponents to play optimally (minimax) this is the result:

Pacman tried to lose as soon as possible!

This strategy is wrong when for example the blue ghost plays randomly, as pacman can gain more points, as shown below:

Question 3:

The evaluation function I implemented takes into account which attributes make a situation better or worse. The attributes I thought of are the score, as a higher score means a better condition, the distance from the nearest dot, since the closer pacman is to a dot the better, the distance (of pacman) from the ghosts and the number of the remaining food and capsules. All these attributes are added to a final sum, which constitutes the evaluation of the situation. Of course, some count negatively in the final evaluation of the situation, such as how many food and capsules are left, that's why we remove these attributes (since if there are many, it means that pacman needs more time to finish the game, so he is in a worse condition). It is noteworthy that to the final sum we add the inverse of closest_food (1 / closest_food) as thus the smaller the distance of closest_food from pacman, the better the evaluation – sum. Also, when the ghosts are very close to the pacman (at manhattan distance less than or equal to 2 – this number was experimentally determined as it had the best score), then**not**we focus on the closest food (for this we set it to infinity, so that the 1 / closest_food added to the final evaluation is almost 0).

Running the command:

```
python pacman.py-pAlphaBetaAgent -aevalFn=better,depth=2-l smallClassic
-k2
```

and comparing her result with that of exercise 1 (minimax), running:

```
python pacman.py-pAlphaBetaAgent -adepth=2-l smallClassic -k2
```

we notice that the agent is more "optimistic" and does not stay in one point as much using the better evaluation function. This behavior is explained as it takes into account more dynamic elements (such as score and distance to the nearest food), unlike minimax which simply relies on the opponent playing perfectly (so they don't make many moves).

So by using the better evaluation function we get a higher score than minimax. It is also notable that the opponents are non-optimal (they make random moves) so the minimax is not optimal.