Doruk ÖZER *70192*

Question 1

I used position of foods, position of pacman ,position of ghosts,timers of ghosts, game score as features. I used position of pacman in order to calculate the manhattan distance between pacman and the closest unscared ghosts, manhattan distance between pacman and the closest food. I used game score in my evaluation to provide some relativeness between states as well as it also decreases as a function of time. I used timers of ghosts in order to detect the unscarred ghosts and use them in the manhattan distance calculations of the closest unscared ghost. The trick I used for punishment mechanism for the closest unscared ghost is as follows: I check if the closest unscared ghost if is in 3 manhattan distance or more; if so I punish the score (subtract from the score) by 55 and if the closest unscared ghost is in less then 3 manhattan distance I punish 65 for 2 manhattan distance, 75 for 1 manhattan distance, 100 for 0 manhattan distance. I use "-" values for the punishments for the score evaluation for the closest manhattan distance between unscared ghost and pacman and closest manhattan distance between food and pacman."-" values of these features is necessary for my implementation since we want to maximize the score because those features are negatively correlated with the score. In addition, if the ghost is scarred code, doesn't punish the factor of closest manhattan distance of unscared ghosts by a simple if statement.

Example of “-” values: as the distance to closest food increase the score of my evaluation decrease and the goal is to maximize the score so pacman should go to closest food so "-" value is preferred. Same logic for manhattan distance of unscared ghost. As the distance of the unscarred ghost decreases the punishment increases so it punishes the score by 75 if the manhattan distance is 1, 65 if manhattan distance is 2.

Question 2

Implementation of MinimaxAgent is much more slower than AlphaBetaAgent. The reason why AlphaBetaAgent is faster is because it prunes the branches which won't affect the final result; so AlphaBetaAgent reduces the exploration space.

Question 3

If the ghosts behaves exactly the same then the outcomes for AlphaBetaAgent, and MinimaxAgent would be exactly the same, because itAlphaBetaAgent prunes the branches which won't affect the final result. So yes the ghosts acted same so the outcome for those two pacmans were the same.

Question 4

The speed of the code of ExpectimaxAgent is same as MinimaxAgent since it nearly makes the same computations (just slight difference for the "minimum layer"). Instead of computing the minimum between the current value and new value for the min layer in the MinimaxAgent, ExpectimaxAgent adds up the expected outcome as final result. So it makes it's computation same as MinimaxAgent.

Question 5

Yes I did write a better evaluation function by just adding a new feature to already existing evaluation function in question 1 (all of the implementations are the same except the new feature and it's evaluation for the score of the current state) .First thing I changed since I am evaluating the current state's evaluation function, is that I evaluated the current game's state features instead of successor game's state features. Secondly and more importantly, I used the position of capsules, I added an award mechanism that awards the pacman if there is a capsule in 2 manhattan distance to pacman and the award increases as the pacman goes near to the capsule and eats it. The implementation logic is exactly the same as I did for the closest unscared ghost but instead, I added the "+" values of this feature since I want to eat the capsule. After pacman eats the capsule it doesn't worries about the position of the ghost anymore so I left the pacman eating the ghost as probabilistic, but it usually eats the ghost.

Question 6

My coefficient array for manhattan distance between pacman and closest unscared ghost: [100: for 0 manhattan distance, 75 for 1 manhattan distance, 65 for 2 manhattan distance, 55 for 3 or more manhattan distance]. I also multiply those values by -3 since it is the feature with the most priority. The final value is added to the current score (if the ghost is scared).

My coefficient array for manhattan distance between pacman and closest capsule: [100: for 0 manhattan distance, 75 for 1 manhattan distance, 65 for 2 or more manhattan distance] I multiply those values by +1 to award the current score if the pacman goes and eats the capsule if the capsule is in the range of 2 manhattan distance in ascending order. I add the final result to the current score. It is the feature with the second priority.

My coefficient for the closest food is -1/12. It is -1/12 because if I made it just -1 then sometimes the pacman stopped and didn't eat the food and as I divide it by value the pacman started stopping less and started searching more for closest food. So I found -1/12 with trial and error. I multiply the closest food by -1/12 and add it to the current score. It is the least prior but most important feature since it is always seeking the closest food. Other features coefficients' depends on the range of those features.