Αντωνης Καλαμακης -Antonis Kalamakis SDI1800056

for evaluationFunction:
i calculated the distance of the food with manhattanDistance and found the closest food. because of the fact that the game keeps going for one last round if you have eaten all the food i had to check if the list that i store the food is empty or not. if yes i just return the default score, if not i calculate the closest food value and i add it to my score-bonus counter. after that i found useful to see if any ghosts are near and if they are scared. if they are scared that means that we are near a capsule or that we have eaten a capsule. so for the action that is extra points! after that for every ghost i have to find if they are really near pacman, if they are we need to immediately abort the action because next frame could be lethal. Decided to use the antistrofh of plus because it is more precise than the normal value.

for minimax:
for my implementation the minimax function is a controller used from min_step and max_step to calculate the next step(we have to calculate the min or max next) and because of that i found useful to see there if the game is done or if its time to get the utility value of the node.

first of all i check if the game is done or utility value needs to be returned.
for (depth == self.depth * gameState.getNumAgents()) i didnt use gameState.getNumAgents()-1 because i start the search with first max_step and the value 0 for depth is skipped always so we start from 1-numAgents. if game is not done or no utility must be returned i check if the agent id is 0, if yes its max's turn. if agent is != 0 its min's step.

for max_step and min_step: (very similar check code for differences) here there is no need to pass the agent id because its always going to be 0. for every successor action i call the minimax function with a new depth and agent id as the depth +1 mod numAgents because that value will always go from 0 to numAgents and we need that to calculate all the depth of min agents. after that i find the maximum value returned and return it as a tuple(thats why at minimax we return the max_Ste()[1] because its the value and not the action).

finally, at the beginning i always start with max_Step because the first user that plays is the pacman so it is the max user.

for alpha_beta:
the same with minimax, but also i have added as arguments the a,b so they will be used inside max_step and min_step.

for max_step:
i do the same with minimax's max and min and after that i check if i need to update the best value, if yes i do it and then i check if its greater than b, that means that we dont need to check for the next actions. if its not i check if its greater than a so i can update the a value for the next alpha_beta function call.

same thing with min_step but with the correct symbols and flipped a with b and b with a.

for expectimax:
for exceptixmax and max_step nothing really changed

for minchance_step:
because of the fact that the probability is the same for every action we just calculate the 1.0/LegalActions to find it. after that we do find the exceptimax for the successor and then we calculate the average value. for the calculation i found that every time it returns a tupple but the last time it just returns a float so i had to use isinstance() function to find that.

for betterEvaluationFunction:
the outcome comes by checking everything around the pacman and generally the status of the game. first of all, of course we have to check if the game is a winner or a looser, the numbers added to plus here are random. after that its wise to check about the closest food available, then the closest capsule and then the closest ghost. because of the fact that a capsule is far more useful than a normal food, if its closer we should think that its wiser to take it so we multiply it by 2 and then add it again. after that the final check we need to do is to see if near us is a ghost, its its next to us we need to l e a v e so we return - infinite so we defiantly wont choose that action. else if its close but not next to us we shouldnt take it as our first choice but it isnt a really bad one either, finally if not then as normal we add it to plus. then to have a more useful output i used the score and added the antistrofo of plus.