Student Name: **A Z Hasnain Kabir**
Student ID: **200042102**

CSE 4618
Artificial Intelligence Lab
Lab 4

Introduction

In this lab, we are to implement MultiAgent Search in pacman.

Question 1

We will improve *ReflexAgent* in *multiAgents.py*.

Analysis of the problem

We need to implement *ReflexAgent* class using out own evalutation function inside *multiAgents.py*.

Explanation of the solution

In our solution, we extract pieces of information regarding the position of our pacman, the position of ghosts and position of foods. Using the Manhatten distance metric, we calculate the minimum distance to any ghost, using the same metric we calculate minimum distance to foods. If the ghost is too close, we return negative infinity to avoid getting too close to ghosts. Thus the best actions are chosen.

Question 2

We will design an adversarial search agent in *MinimaxAgent* class in *multiAgent.py*

Analysis of the problem

We need to write code to implement minimax search where there will be one max node for pacman and multiple min nodes (each one for every ghost).

Explanation of the solution

We determine agent indices for Pacman being 0 and ghosts greater than 0. Then we implement a recursive function that calculates minimum and maximum values for each of the successor nodes. The algorithm runs recursively to find the actual minimum and maximum values and returns one value at the very top.

Question 3

Another technique called alpha-beta pruning needs to be used in minimax search.

Analysis of the problem

We implement alpha-beta pruning inside *AlphaBetaAgent* class in *MultiAgents.py* for minimax search alpha-beta pruning.

Explanation of the solution

The implementation is quite similar to minimax search except in two places. For instance, in maximum value calculation, we define a logic where if the current value in any state is greater than beta then we return the values at hand or else we need to maximize alpha between the current value and itself. On the other hand, in minimum value calculation, we define the opposite logic for alpha.

Question 4

We need to implement Expectimax search for pacman.

Analysis of the problem

We implement *ExpectimaxAgent* class inside *multiAgents.py* for Expectimax search.

Explanation of the solution

The solution is quite similar to minimax search implementation except in nodes which needs expectimax to return correct values, we define a function that calculates probability of each node by taking the reciprocal of total number of actions. Our function then uses that probability to calculate the return value of each node.

Question 5

We need to design a better evaluation function for pacman.

Analysis of the problem

We need to implement the function *betterEvaluationFunction* from *multiAgents.py*. It should evaluate states rather than actions.

Explanation of the solution

We start by extracting essential information like position of pacman, the remaining food pallets and the ghosts. One penalty is defined for being too close to ghost, two rewards defined for being too close to food pallets and scared ghosts. The reward for being close to scared ghost is kept high. We calculate minimum distance to any food pallets using manhattan distance and update the score by adding reward if pacman is close to a food pallet. The reward is the inverse of distance to any food pallet so the closer pacman will be the more rewards will be gained. It calculates the

minimum distance of pacman to ghosts and rewards or penalizes based on the state of the ghosts. If the ghosts are scared, it gets a very high reward and if the ghost is not scared, it is penalized. The calculated score is returned from the evaulation function.