CSE 537 Artificial Intelligence	Assignment 02
Gaurav Gawde	110008637

----------Question 1----------

ReflexAgent Class:

Changes are made to the ReflexAgent Class to play respectably.A capable reflex agent should consider both food locations and ghost locations to perform well.Reflex agent will die when there are 2 ghosts on the board and when evaluation function is not good.

Following formula is used to improve the efficiency of the evaluation function in ReflexAgent Class.

Initial Evaluation Function Return Value:
successorGameState.getScore()

Improved Evaluation Function Return Value:
successorGameState.getScore()+(food_score*ghost_distance)+ghost_scared_score

Explanation:
I used different formulas for getting good evaluation function.But above formula gave best efficiency and hence it is used.

Parameter Details:
food_score=To consider food locations while making a decision
ghost_distance=To consider ghost distance while making a decision
ghost_scared_score=To consider whether ghost is scared or not after taking power pellet

----------Question 2----------

MinimaxAgent Class:

minimax agent is created which will work with any number of ghosts.minimax tree is created which has multiple min layers for each max layer.

Below are the functions created to achieve above functionality:

find_max_val():
This function returns maximum value from set of minimum values for pacman agent.

find_min_val():This function returns minimum value from set of maximum values for ghost agent.

leaf_node():
Checks whether you have reached leaf node of pacman agent.

ghost_leaf_node():
Checks whether you have reached leaf node of ghost agent.

self.evaluationFunction:Used for scoring leaves of the minimax tree.

----------Question 3----------

AlphaBetaAgent Class:

Methods are added to AlphaBetaAgent class to efficiently explore the minimax tree using alpha-beta pruning.Alpha-beta pruning logic is extended efficiently to multiple minimizer agents.

Following functions are created to achieve required functionality:

find_max_val():
This function finds maximum values based on gamestate,depth,alpha and beta values.

find_min_val():
This function finds minimum values based on gamestate,agent_index,depth,alpha and beta

----------Statistics----------

Question 1:

```
Command:pacman.py
```

Output:

Time Required To Execute The Program:: 2.98435211182 seconds
Memory Taken By The Program(KBytes):: 705588 KB
Number Of Nodes Expanded:: 60

```
Command:python pacman.py -p ReflexAgent
```

Output:

Time Required To Execute The Program:: 0.41416311264 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 716

Command:`python pacman.py -p ReflexAgent -l testClassic`

Output:

Time Required To Execute The Program:: 0.0162041187286 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 60

Command:`python pacman.py --frameTime 0 -p ReflexAgent -k 1`

Output:
Time Required To Execute The Program:: 0.395692825317 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 854

Command:`python pacman.py --frameTime 0 -p ReflexAgent -k 2`

Output:

Time Required To Execute The Program:: 0.700345039368 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 1117

Command:`python pacman.py -p ReflexAgent -l openClassic -n 10 -q`

Output:

Pacman emerges victorious! Score: 983
Pacman emerges victorious! Score: 1146
Pacman emerges victorious! Score: 1253
Pacman emerges victorious! Score: 1243
Pacman emerges victorious! Score: 971
Pacman emerges victorious! Score: 1153
Pacman emerges victorious! Score: 1208

Pacman emerges victorious! Score: 1102
Pacman emerges victorious! Score: 1224
Pacman emerges victorious! Score: 1246
Average Score: 1152.9
Scores: 983.0, 1146.0, 1253.0, 1243.0, 971.0, 1153.0, 1208.0, 1102.0, 1224.0, 1246.0
Win Rate: 10/10 (1.00)
Record: Win, Win, Win, Win, Win, Win, Win, Win, Win, Win

Time Required To Execute The Program:: 3.81039905548 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 9079

Question 2:

Command:python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4

Output:

Time Required To Execute The Program:: 1.68995499611 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 5

Command:

python pacman.py -p MinimaxAgent -l trappedClassic -a depth=3

Output:

Time Required To Execute The Program:: 0.0135929584503 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 1

Question 3:

Command:

```
python pacman.py -p AlphaBetaAgent -a depth=3 -l
smallClassic
```

Output:

Time Required To Execute The Program:: 4.53047680855 seconds

Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 60

----------Critical Analysis----------

(A)

Command:

```
python pacman.py -p AlphaBetaAgent -a depth=3 -l
smallClassic
```

Output:
Time Required To Execute The Program:: 4.53047680855 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 60

(B)

```
python pacman.py -p MiniMaxAgent -a depth=3 -l smallClassic
```

Time Required To Execute The Program:: 46.1357619762 seconds
Memory Taken By The Program(KBytes):: 705700 KB
Number Of Nodes Expanded:: 490

Explanation:

AlphaBetaAgent works efficiently for depth=3 and on smallClassic board as compared to MiniMaxAgent for depth=3 and on smallClassic board as shown above.Thus we can conclude that alpha-beta pruning done in AlphaBetaAgent class saves a lot of time,unnecessary node expansion and memory space.