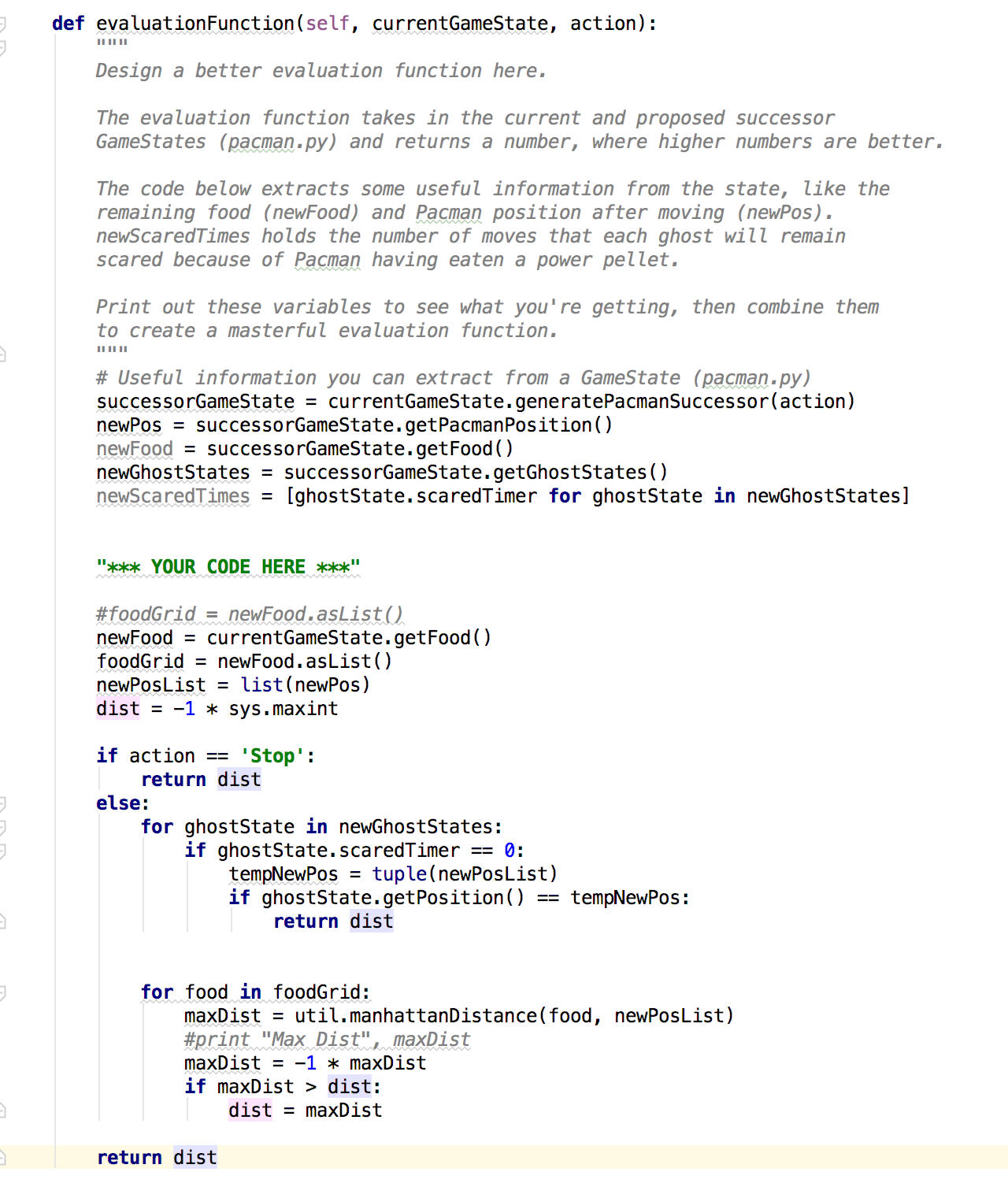
**HW2 Project Details Document**

1. ***Question 1***Reflex Agent:

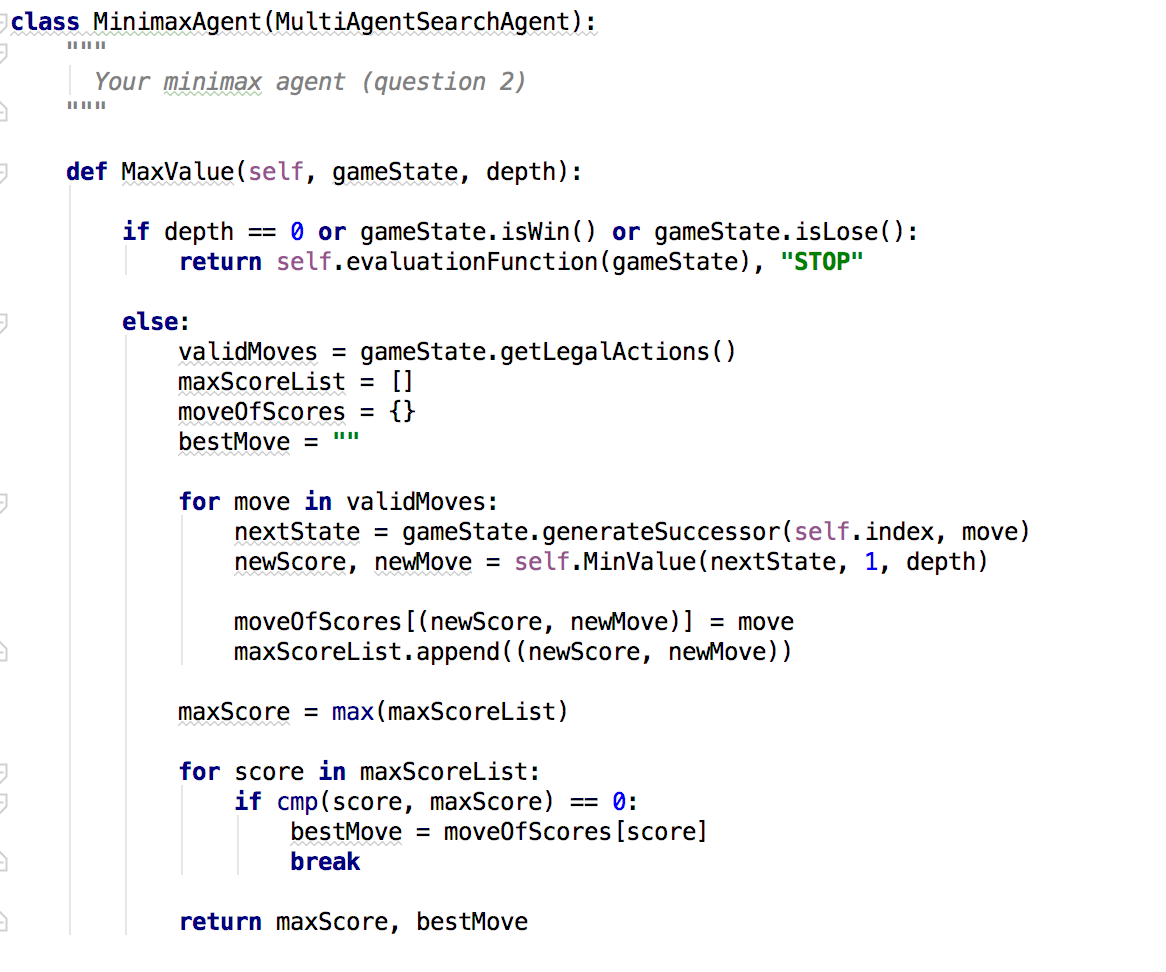
* **Code Details:**

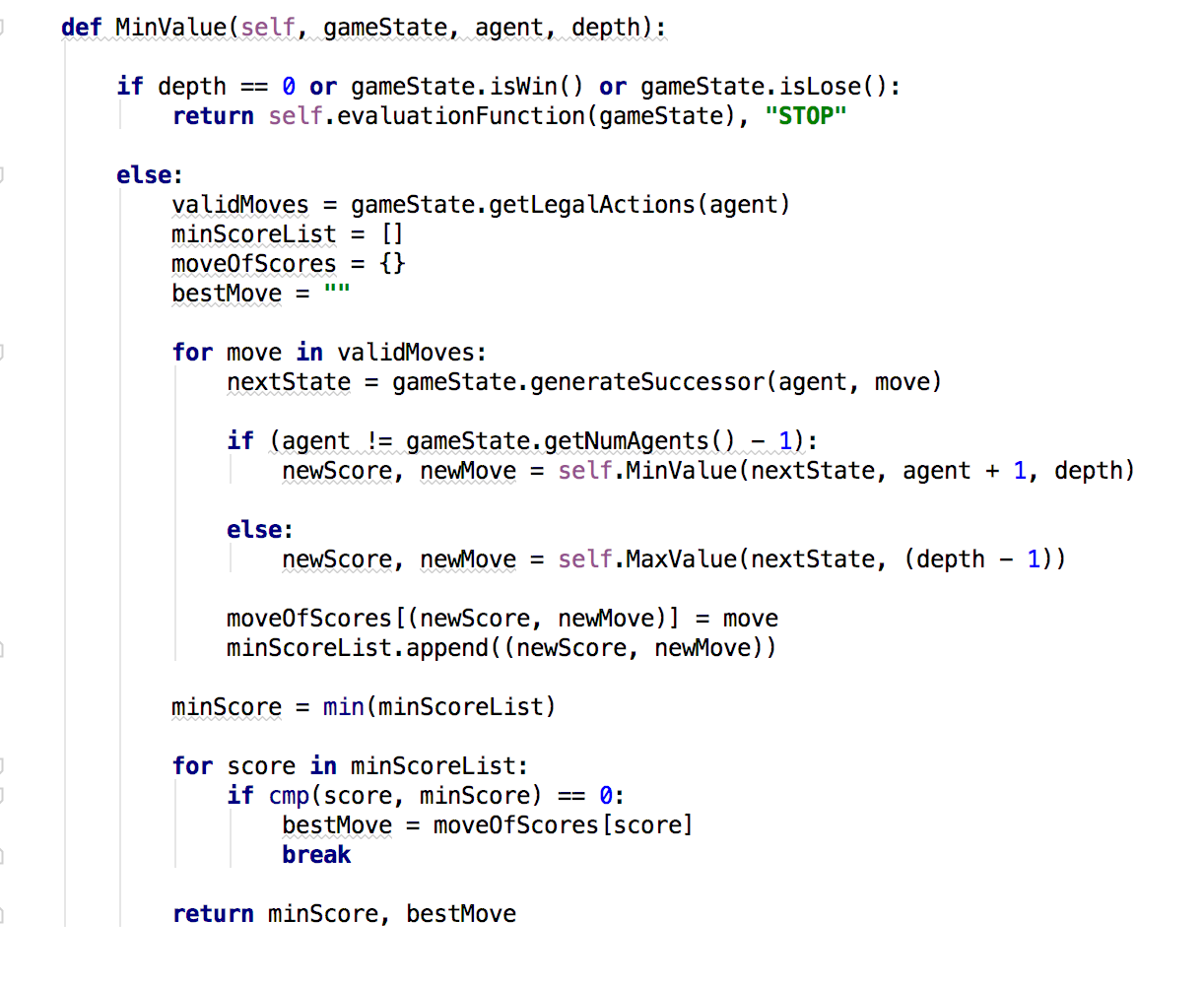


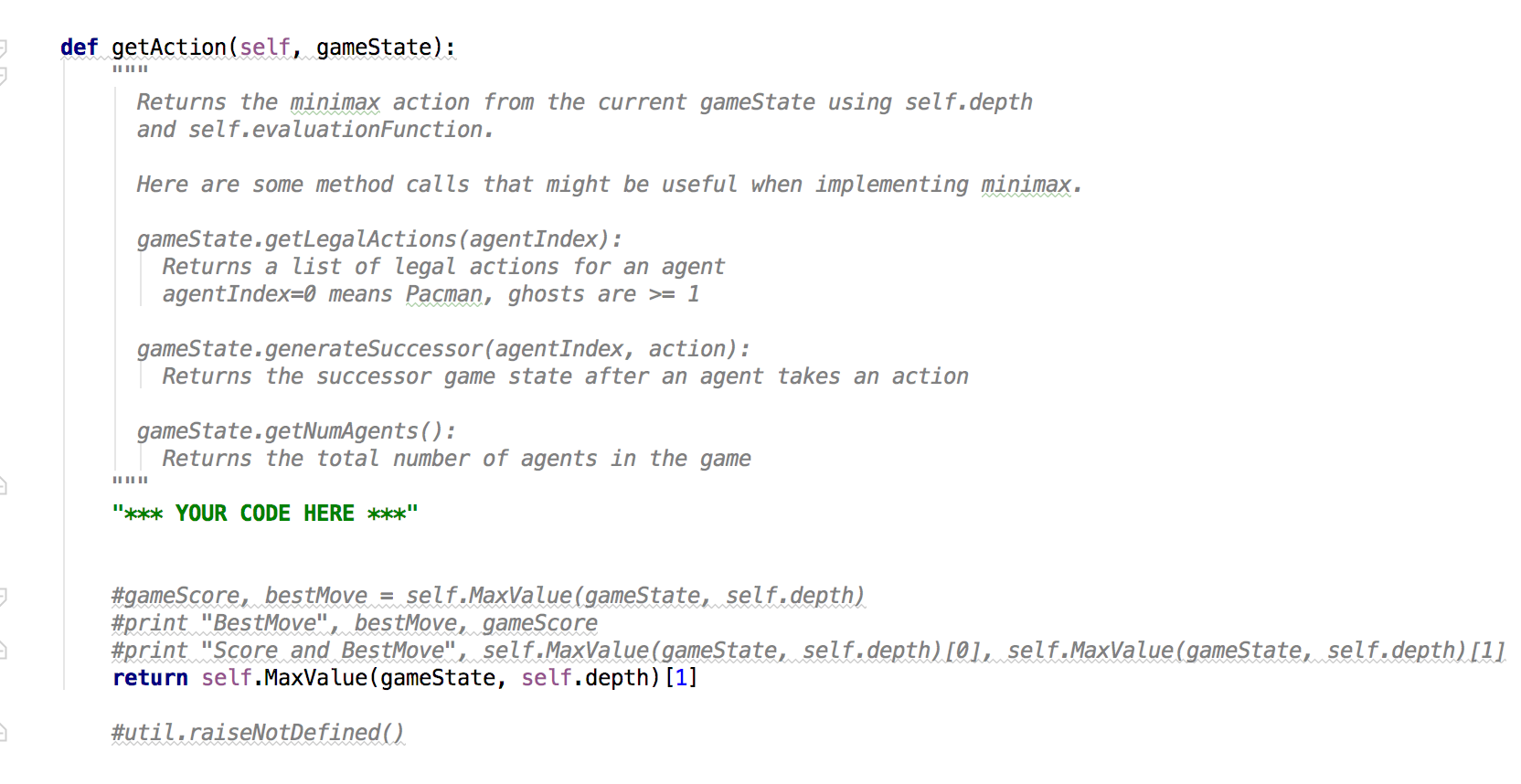
* I have updated the code inside evaluationFunction() method in ReflexAgent class.
* We will generate the successor nodes/states using the current game state of Pacman.
* Find all the positions of Pacman
* For the successor nodes get all the ghost states and find the scared timer for all the ghost states.
* Using the game state gate all the food positions of Pacman and along its movement path.
* Convert the food as a food grid list.
* Convert the Pacman’s positions as a list.
* Initialize the distance with minimum value
* If input action is ‘Stop’ the distance value
* Else loop over all the ghost states:
  + If scared timer of ghost is 0 and position of ghost state is same with Pacman position then in that case return the minimum distance.
  + Loop over all the foods in the food grid:
    - Calculate manhattan distance between food location and Pacman’s position
    - Make this new distance value to negative to compare this with previous distance value. So that we can get the minimum distance.
    - Compare new distance and previous distance and set the new distance accordingly
* Return the calculated distance which will be used by the game to move the Pacman accordingly.

1. **Question 2**Minimax:

* **Code Details:**



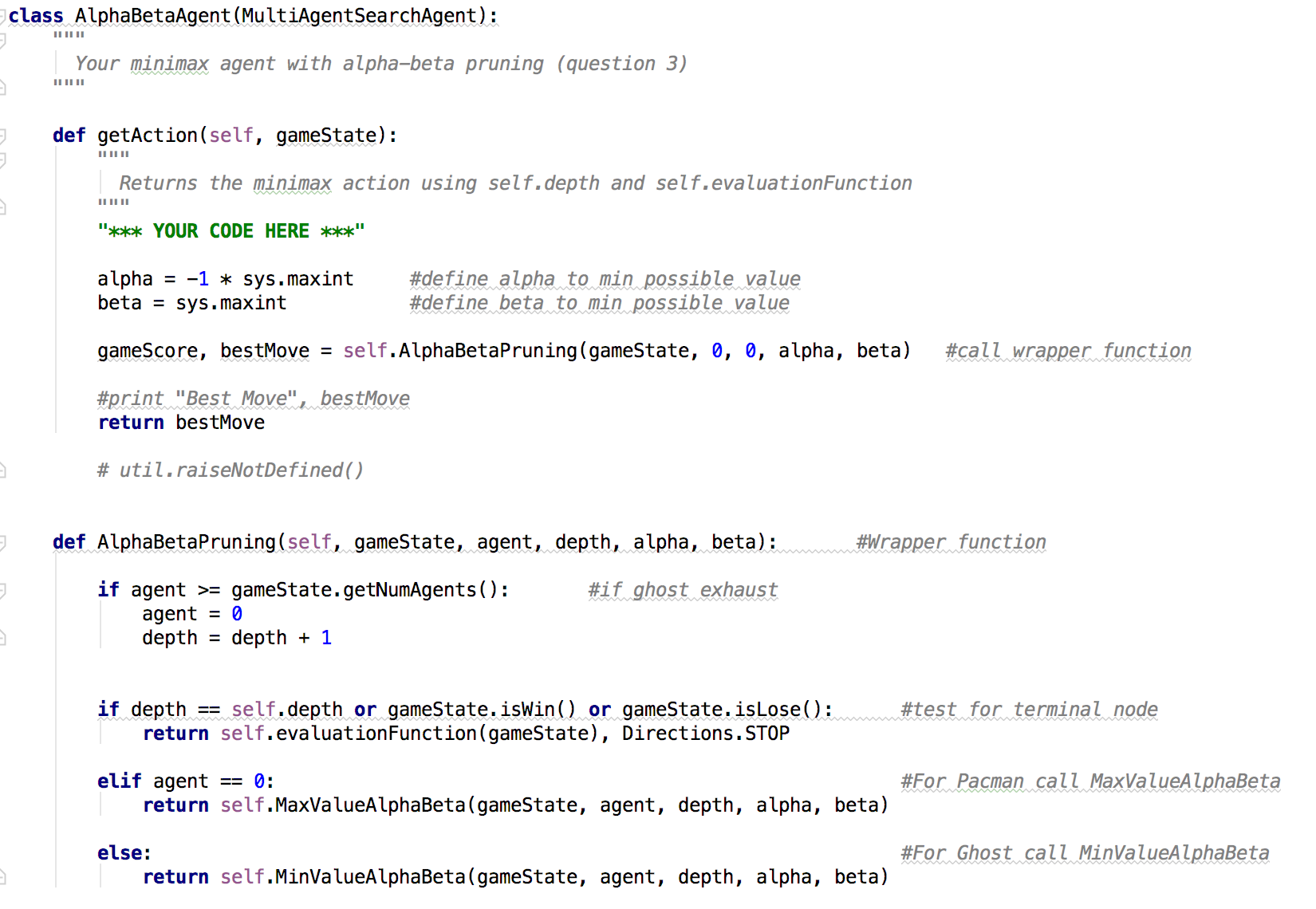


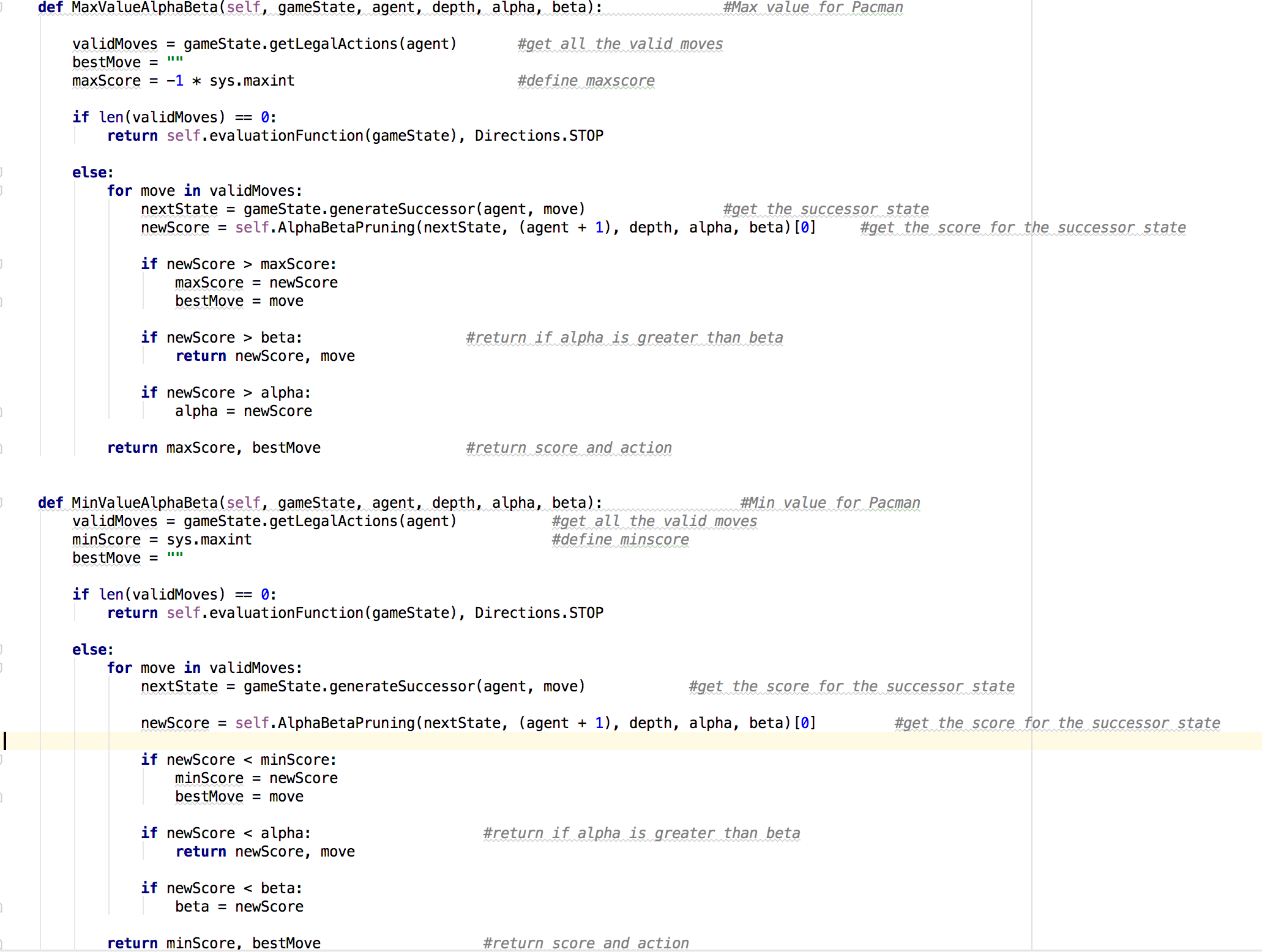


* I have developed 3 methods MaxValue(), MinValue() and getAction() method as part of MiniMaxAgent.
* MaxValue() implementation:
  + If depth is 0 or node is terminal node then return the score value of that state and move(“STOP”) from that step
  + If not terminal state then get all the valid actions/moves for the current state.
  + Initialize maxScoreList list, moveOfScore dict and bestMove variables.
  + For each move in valid moves:
    - Generate the successor of the current state
    - Call MinValue() function with successor state, agent value and game depth value. This will return state score and best move
    - Append the retrieved values to moveOfScore dictionary as key and set the current move as the value.
    - Append the retrieved values to maxScoreList list.
  + Retrieve the max score from maxScoreList list.
  + Loop over each score of maxScoreList list:
    - Get the first repetition of max score in the list.
    - For the current score retrieve the move from the moveOfScore dictionary.
    - Assign this value to bestMove.
  + Return the max score and bestMove from this method call.
* MinValue() implementation:
  + If depth is 0 or node is terminal node then return the score value of that state and move(“STOP”) from that step
  + If not terminal state then get all the valid actions/moves for the current state.
  + Initialize minScoreList list, moveOfScore dict and bestMove variables.
  + For each move in valid moves:
    - Generate the successor of the current state
    - If more ghosts are present then call the MinValue() method recursively and pass the successor state, agent and depth value.
    - Else all the ghosts are evaluated then call the MaxValue() method and pass successor state and depth value.
    - Capture the retrieved values(score and move) in the new variables.
    - Append the retrieved values to moveOfScore dictionary as key and set the current move as the value.
    - Append the retrieved values to minScoreList list.
  + Retrieve the min score from minScoreList list.
  + Loop over each score of minScoreList list:
    - Get the first repetition of min score in the list.
    - For the current score retrieve the move from the moveOfScore dictionary.
    - Assign this value to bestMove.
  + Return the min score and bestMove from this method call.
* getAction() implementation:
  + Call the MaxValue() with current game state and depth value
  + Return the best move returned by the method.

1. **Question 3** Alpha-Beta Pruning:

* **Code Details:**

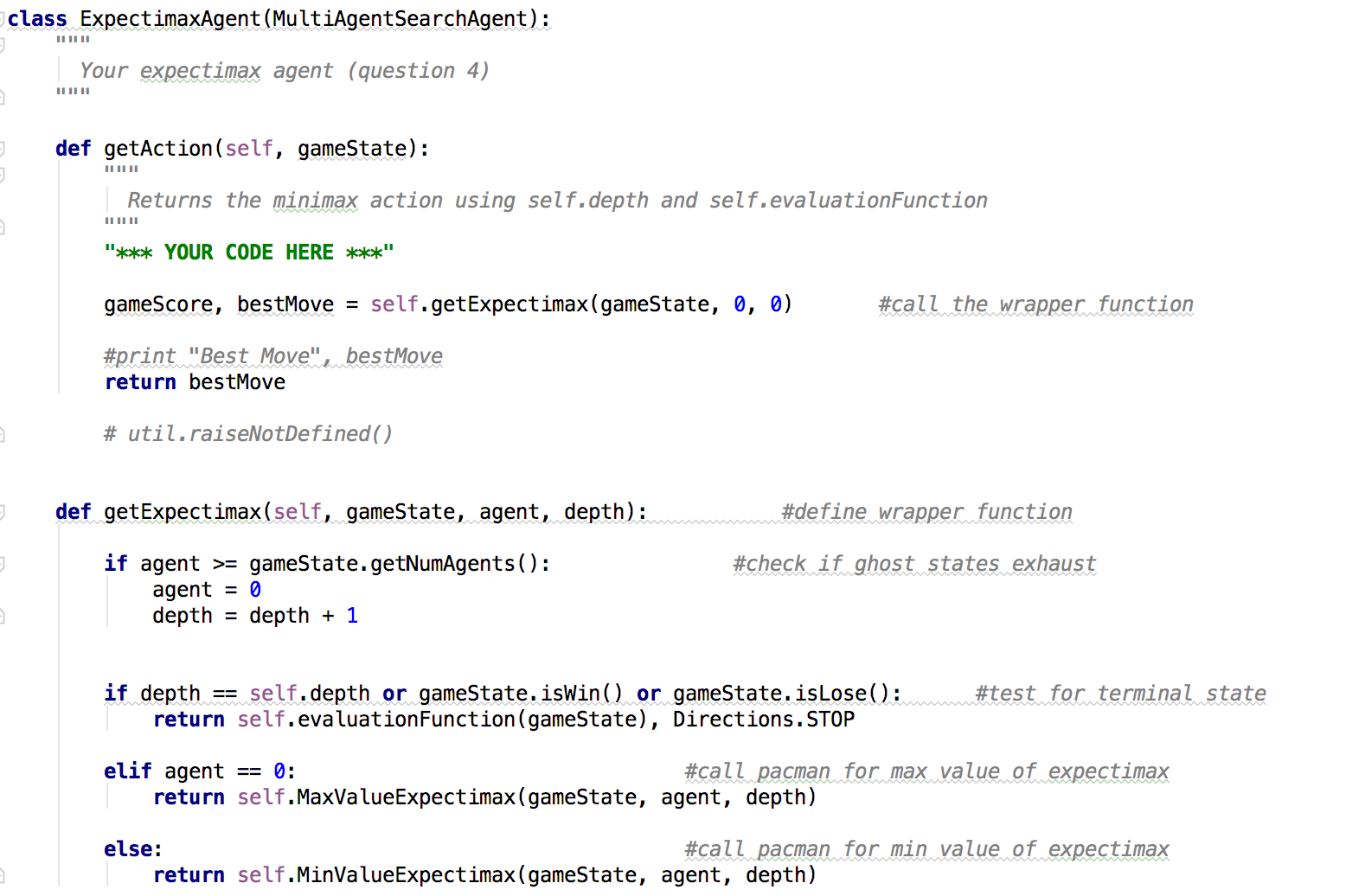


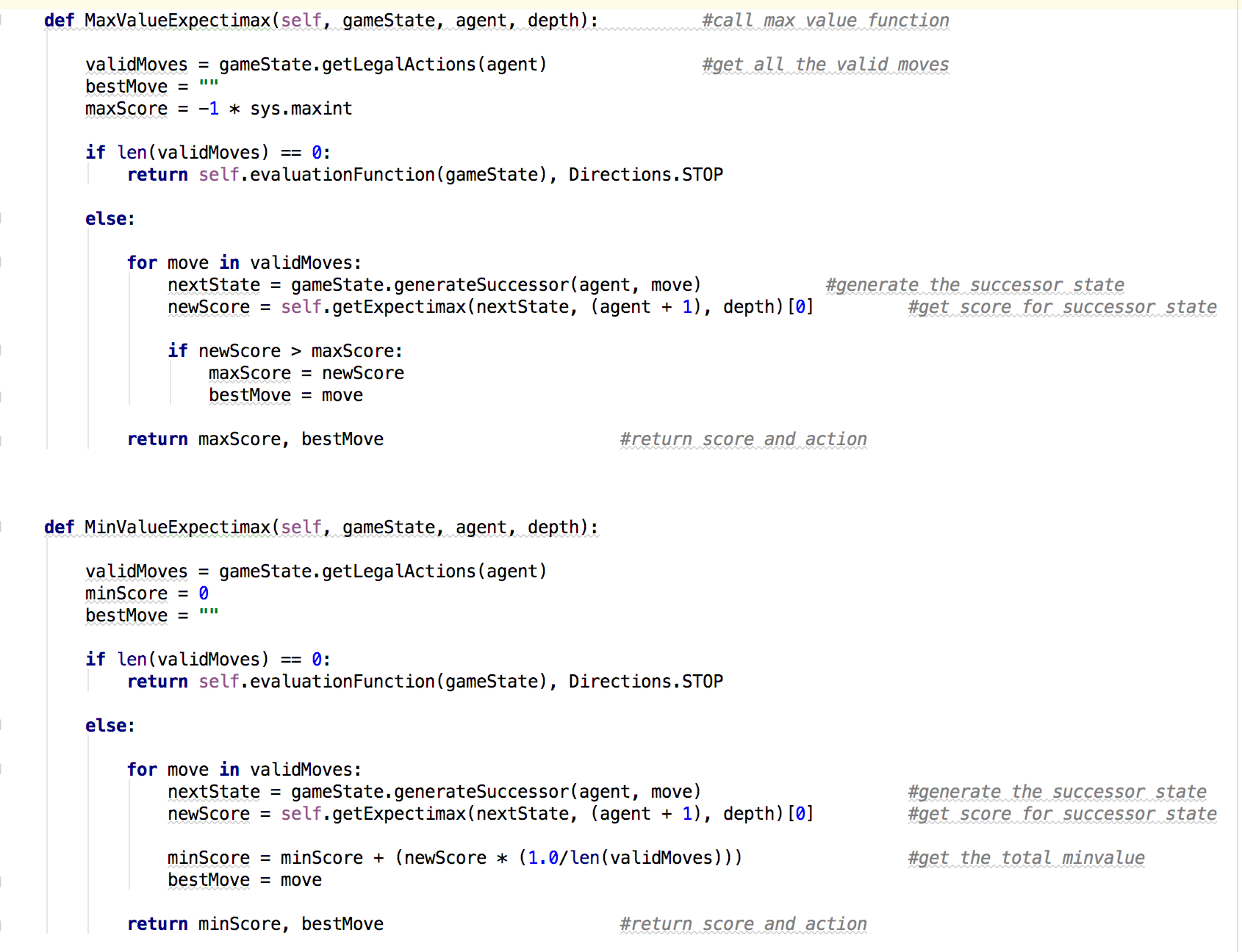


* I have developed 4 methods AlphaBetaPruning(), MaxValueAlphaBeta(), MinValueAlphaBeta() and getAction() method as part of AlphaBetaAgent.
* AlphaBetaPruning() implementation:
  + Check for agent value is greater than the gamestate agents numbers. This will check if the ghosts are exhausted or not.
  + If ghost exhausted then set agent value to 0
  + Increase depth by 1
  + If depth is the gamestate depth or state is won or lost then this is the terminal state.
  + If no ghost is present then pacman moves and find the max value
  + Else the ghosts moves and find the min values
* MaxValueAlphaBeta() implementation:
  + Get all the valid actions/moves for the current state.
  + Initialize variables bestMove and maxScore which is highest minimum value.
  + If no valid moves are present then return the value of current state
  + For each move in valid moves:
    - Generate the successor of the current state
    - Call AlphaBetaPruning() function with successor state, agent value game depth value, alpha and beta value. This will return state score
    - If new score is greater than maxscore then:
      * Set the new score value as maxscore.
      * Get the best move from the valid moves.
    - If alpha > beta then return alpha and best move from this method call.
    - Set the newscore to alpha value
  + Return the max score and bestMove from this method call.
* MinValueAlphaBeta() implementation:
  + Get all the valid actions/moves for the current state.
  + Initialize variables minScore which is highest maximum value and bestmove.
  + If no valid moves are present then return the value of current state
  + For each move in valid moves:
    - Generate the successor of the current state
    - Call AlphaBetaPruning() function with successor state, agent value game depth value, alpha and beta value. This will return state score.
    - Capture the retrieved values(score) in the new variable.
    - If new score is less than minscore then set this to new minscore and set the best move to current move
    - If new score is less than alpha then return alpha and move
    - If new score is less than beta then set new score to beta
  + Return the min score and best move from this method call.
* getAction() implementation:
  + initialize alpha as the maximum negative number and beta as the maximum positive number
  + Call the AlphaBetaPruning() with current game state, depth value, alpha and beta values.
  + Return the best move returned by the method call.

1. **Question 4**Expectimax:

* **Code Details:**





* I have developed 4 methods getExpectimax(),MaxValueExpectimax(),MinValueExpectimax() and getAction() method as part of AlphaBetaAgent.
* getExpectimax() implementation:
  + Check for agent value is greater than the gamestate agents numbers. This will check if the ghosts are exhausted or not.
  + If ghost exhausted then set agent value to 0
  + Increase depth by 1
  + If depth is the gamestate depth or state is won or lost then this is the terminal state.
  + If no ghost is present then pacman moves and find the max value
  + Else the ghosts move and find the min values
* MaxValueExpectimax() implementation:
  + Get all the valid actions/moves for the current state.
  + Initialize variables bestMove and maxScore which is highest minimum value.
  + If no valid moves are present then return the value of current state
  + For each move in valid moves:
    - Generate the successor of the current state
    - Call getExpectimax() function with successor state, agent value game and depth value. This will return state score
    - If new score is greater than maxscore then:
      * Set the new score value as maxscore.
      * Get the best move from the valid moves.
  + Return the max score and bestMove from this method call.
* MinValueExpectimax() implementation:
  + Get all the valid actions/moves for the current state.
  + Initialize variables minScore which is 0 value and bestmove.
  + If no valid moves are present then return the value of current state
  + For each move in valid moves:
    - Generate the successor of the current state
    - Call getExpectimax() function with successor state, agent value game and depth value. This will return state score.
    - Capture the retrieved values(score) in the new variable.
    - Calculate minscore as oldminscore value + (1.0/length of valid moves)
    - set the best move to current move
  + Return the min score and best move from this method call.
* getAction() implementation:
  + Call the getExpectimax() with current game state, agent and depth value
  + Return the best move returned by the method call.