Miguel Angel Nivia Ortega
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

21/09/2024

NOTA: Para la realización del trabajo se usó Claude(IA) e hicimos los siguientes prompts para llegar a las respuestas.

PROMPTS REALIZADOS

Con base en el proyecto berkeley sobre la implementación de algoritmos de búsqueda para el juego de pacman, construido en Python 2.7.0. A continuación, se detallan las instrucciones para resolver los problemas de Q1 a Q5:

Q1:

Instrucciones

Debes implementar en el algoritmo de multiAgents.py en la clase de ReflexAgent el código para que pacman, logre de forma satisfactoria la recolección de la comida, sin ser atrapado por los fantasmas, en un tiempo rápido y de manera eficaz.

1. Implementar el algoritmo:

- Implementa el algoritmo en la clase ReflexAgent.
- El algoritmo debe retornar el mejor puntaje obtenido en base a los movimientos legales que pueda hacer, para esto se debe tener en cuenta el sucesor y la posición actual del mismo, además de saber la posición de los fantasmas y si estos o no pueden lograr matar a pacman para evitarlos o lograr conseguir la fruta especial para sobrevivir.

2. Uso extras:

- De ser necesario utiliza funciones que se encuentran dentro del mismo código o de los imports.

Consideraciones:

- **Solo puedes modificar en la clase de ReflexAgent.**

Codigo Inicial:

```
class ReflexAgent(Agent):
  """
   A reflex agent chooses an action at each choice point by examining

   its alternatives via a state evaluation function.

   The code below is provided as a guide.  You are welcome to change

   it in any way you see fit, so long as you don't touch our method

   headers.
  """

  def getAction(self, gameState):
    """

    You do not need to change this method, but you're welcome to.


    getAction chooses among the best options according to the evaluation function.

    Just like in the previous project, getAction takes a GameState and returns
```

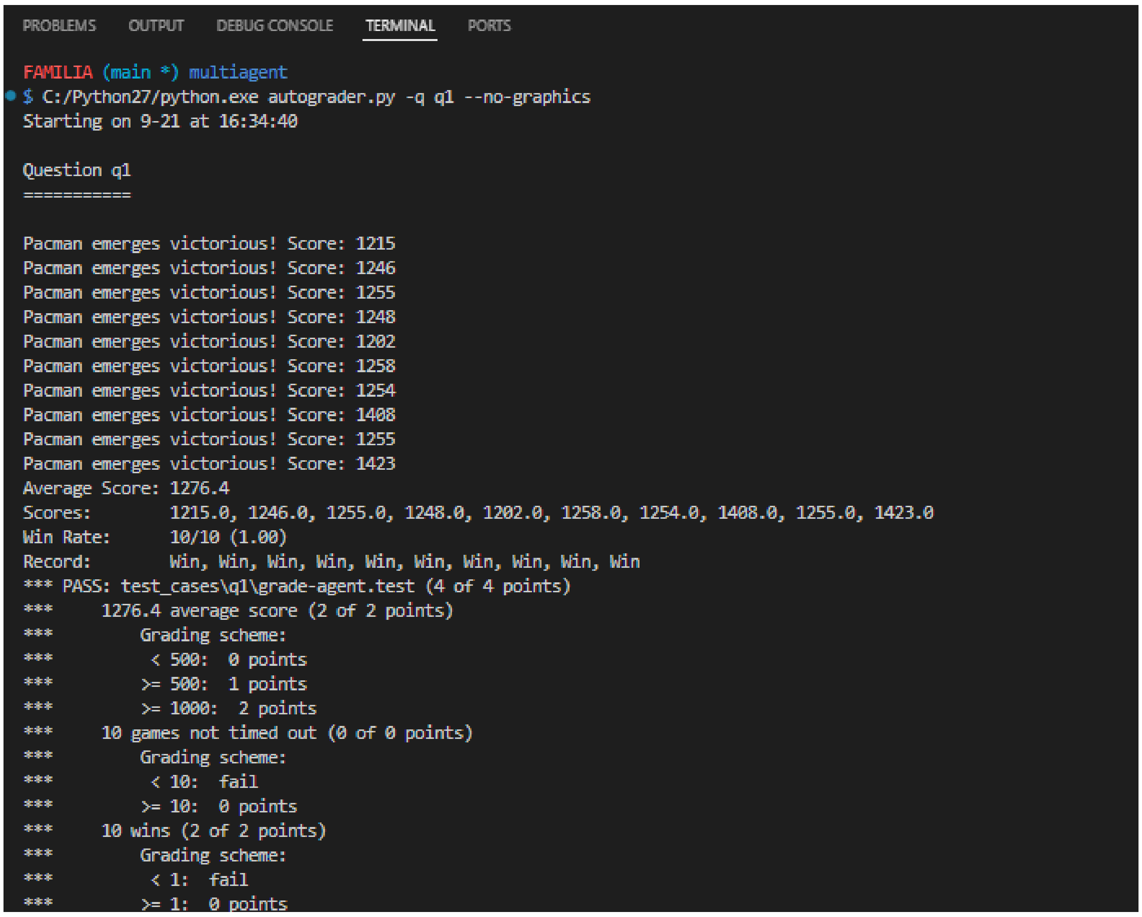
```
        some Directions.X for some X in the set {North, South, West, East, Stop}
        """
        # Collect legal moves and successor states
        legalMoves = gameState.getLegalActions()
        # Choose one of the best actions
        scores = [self.evaluationFunction(gameState, action) for action in legalMoves]
        bestScore = max(scores)
        bestIndices = [index for index in range(len(scores)) if scores[index] == bestScore]
        chosenIndex = random.choice(bestIndices) # Pick randomly among the best
        "Add more of your code here if you want to"
        return legalMoves[chosenIndex]

    def evaluationFunction(self, currentGameState, action):
        """
        Design a better evaluation function here.
        The evaluation function takes in the current and proposed successor
        GameStates (pacman.py) and returns a number, where higher numbers are better.
        The code below extracts some useful information from the state, like the
        remaining food (newFood) and Pacman position after moving (newPos).
        newScaredTimes holds the number of moves that each ghost will remain
        scared because of Pacman having eaten a power pellet.
        Print out these variables to see what you're getting, then combine them
        to create a masterful evaluation function.
        """
        # Useful information you can extract from a GameState (pacman.py)
        successorGameState = currentGameState.generatePacmanSuccessor(action)
        newPos = successorGameState.getPacmanPosition()
        newFood = successorGameState.getFood()
        newGhostStates = successorGameState.getGhostStates()
        newScaredTimes = [ghostState.scaredTimer for ghostState in newGhostStates]
        "*** YOUR CODE HERE ***"
        return successorGameState.getScore()
```

Miguel Angel Nivia Ortega
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

21/09/2024

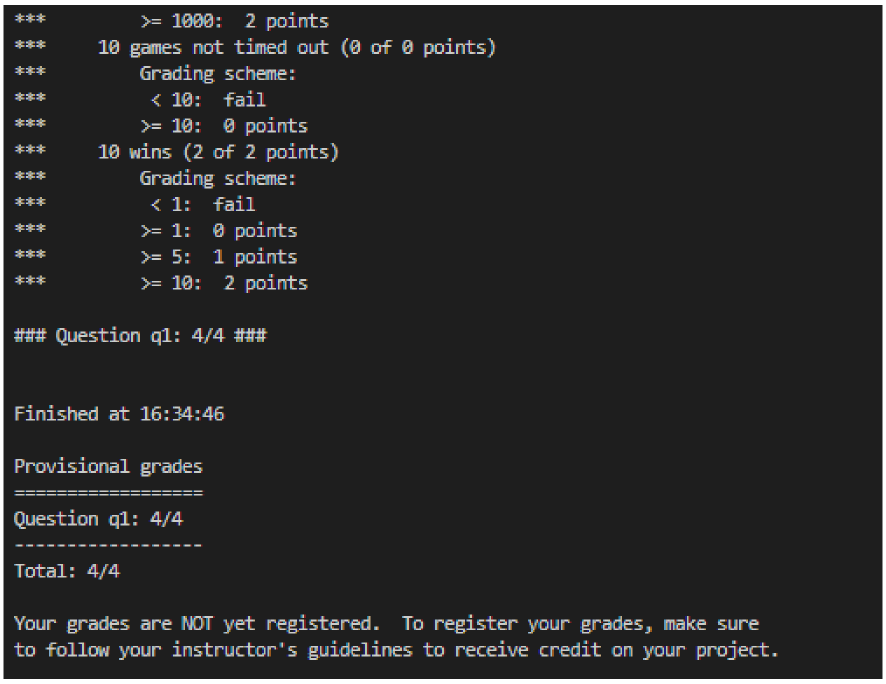
Resultado Autograder:



```
FAMILIA (main *) multiagent
$ C:/Python27/python.exe autograder.py -q q1 --no-graphics
Starting on 9-21 at 16:34:40

Question q1
===========

Pacman emerges victorious! Score: 1215
Pacman emerges victorious! Score: 1246
Pacman emerges victorious! Score: 1255
Pacman emerges victorious! Score: 1248
Pacman emerges victorious! Score: 1202
Pacman emerges victorious! Score: 1258
Pacman emerges victorious! Score: 1254
Pacman emerges victorious! Score: 1408
Pacman emerges victorious! Score: 1255
Pacman emerges victorious! Score: 1423
Average Score: 1276.4
Scores:        1215.0, 1246.0, 1255.0, 1248.0, 1202.0, 1258.0, 1254.0, 1408.0, 1255.0, 1423.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases\q1\grade-agent.test (4 of 4 points)
***     1276.4 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (0 of 0 points)
***         Grading scheme:
***          < 10:  fail
***         >= 10:  0 points
***     10 wins (2 of 2 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  0 points
```



```
***         >= 1000:  2 points
***     10 games not timed out (0 of 0 points)
***         Grading scheme:
***          < 10:  fail
***         >= 10:  0 points
***     10 wins (2 of 2 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  0 points
***         >= 5:  1 points
***         >= 10:  2 points

### Question q1: 4/4 ###


Finished at 16:34:46

Provisional grades
==================
Question q1: 4/4
------------------
Total: 4/4

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

Miguel Angel Nivia Ortega
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

21/09/2024

Q2:

Instrucciones

Debes implementar en el algoritmo de multiAgents.py en la clase MinimaxAgent para hacer la implementación que realice una evaluación de caminos similar al del Q1 pero usando arboles de profundidad para obtener una mejora y rapidez de eficacia de datos para el pacman.

1. Implementar el algoritmo:

- Implementa el algoritmo en la clase MinimaxAgent.
- El algoritmo debe retornar el mejor camino posible que debe recorrer pacman.

2. Uso extras:

- De ser necesario utiliza funciones que se encuentran dentro del mismo código o de los imports.

Consideraciones:

- **Solo puedes modificar en la clase de MinimaxAgent.**

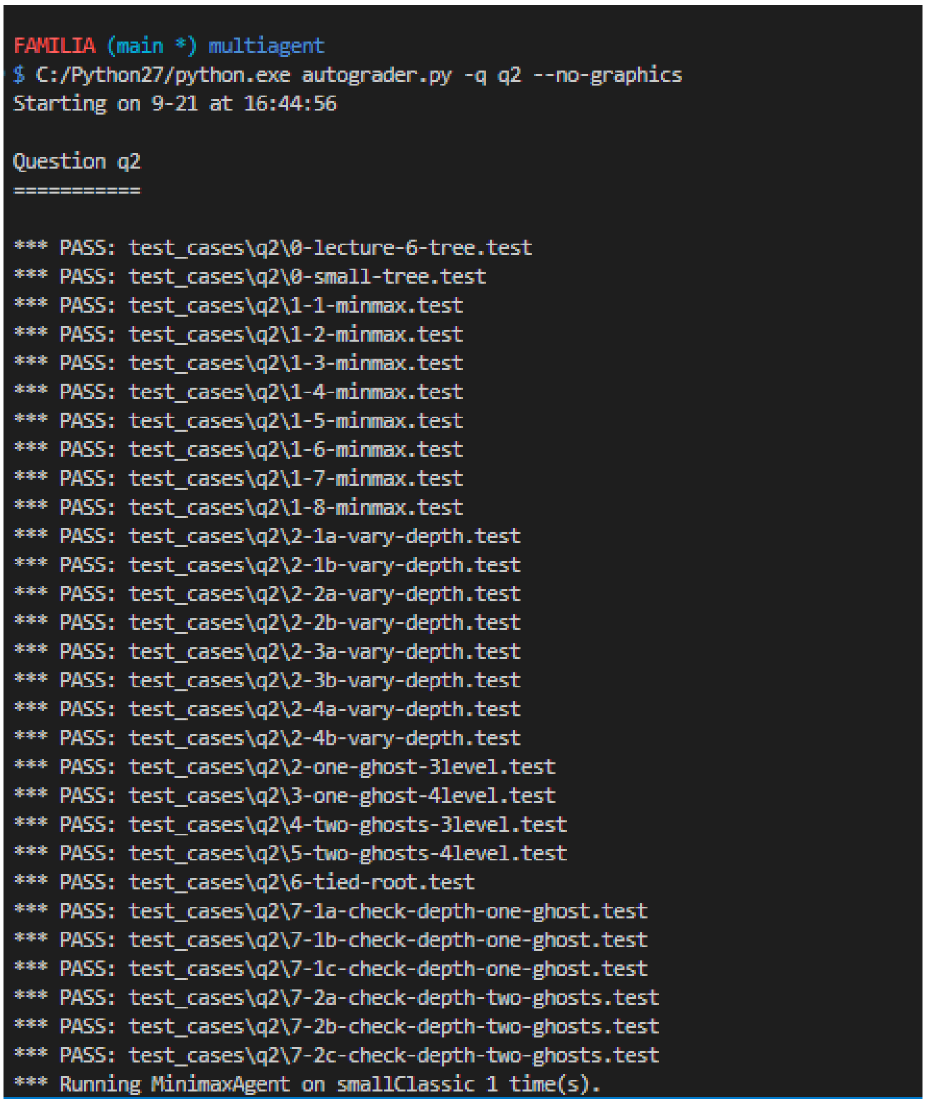
Codigo Inicial:

```
class MinimaxAgent(MultiAgentSearchAgent):
    """
      Your minimax agent (question 2)
    """
    def getAction(self, gameState):
        """
          Returns the minimax action from the current gameState using self.depth
          and self.evaluationFunction.
          Here are some method calls that might be useful when implementing minimax.
          gameState.getLegalActions(agentIndex):
            Returns a list of legal actions for an agent
            agentIndex=0 means Pacman, ghosts are >= 1
          gameState.generateSuccessor(agentIndex, action):
            Returns the successor game state after an agent takes an action
          gameState.getNumAgents():
            Returns the total number of agents in the game
        """
        "*** YOUR CODE HERE ***"
        util.raiseNotDefined()
```

Miguel Angel Nivia Ortega
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

21/09/2024

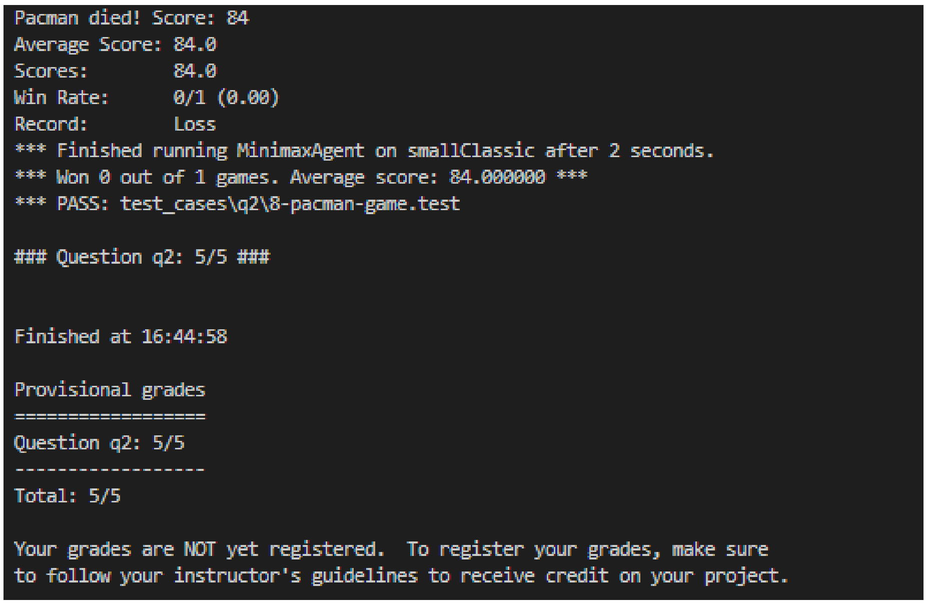
Resultado Autograder:



```
FAMILIA (main *) multiagent
$ C:/Python27/python.exe autograder.py -q q2 --no-graphics
Starting on 9-21 at 16:44:56

Question q2
===========

*** PASS: test_cases\q2\0-lecture-6-tree.test
*** PASS: test_cases\q2\0-small-tree.test
*** PASS: test_cases\q2\1-1-minmax.test
*** PASS: test_cases\q2\1-2-minmax.test
*** PASS: test_cases\q2\1-3-minmax.test
*** PASS: test_cases\q2\1-4-minmax.test
*** PASS: test_cases\q2\1-5-minmax.test
*** PASS: test_cases\q2\1-6-minmax.test
*** PASS: test_cases\q2\1-7-minmax.test
*** PASS: test_cases\q2\1-8-minmax.test
*** PASS: test_cases\q2\2-1a-vary-depth.test
*** PASS: test_cases\q2\2-1b-vary-depth.test
*** PASS: test_cases\q2\2-2a-vary-depth.test
*** PASS: test_cases\q2\2-2b-vary-depth.test
*** PASS: test_cases\q2\2-3a-vary-depth.test
*** PASS: test_cases\q2\2-3b-vary-depth.test
*** PASS: test_cases\q2\2-4a-vary-depth.test
*** PASS: test_cases\q2\2-4b-vary-depth.test
*** PASS: test_cases\q2\2-one-ghost-3level.test
*** PASS: test_cases\q2\3-one-ghost-4level.test
*** PASS: test_cases\q2\4-two-ghosts-3level.test
*** PASS: test_cases\q2\5-two-ghosts-4level.test
*** PASS: test_cases\q2\6-tied-root.test
*** PASS: test_cases\q2\7-1a-check-depth-one-ghost.test
*** PASS: test_cases\q2\7-1b-check-depth-one-ghost.test
*** PASS: test_cases\q2\7-1c-check-depth-one-ghost.test
*** PASS: test_cases\q2\7-2a-check-depth-two-ghosts.test
*** PASS: test_cases\q2\7-2b-check-depth-two-ghosts.test
*** PASS: test_cases\q2\7-2c-check-depth-two-ghosts.test
*** Running MinimaxAgent on smallClassic 1 time(s).
```



```
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running MinimaxAgent on smallClassic after 2 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q2\8-pacman-game.test

### Question q2: 5/5 ###


Finished at 16:44:58

Provisional grades
==================
Question q2: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

Miguel Angel Nivia Ortega **21/09/2024**
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

Q3:

Instrucciones

Debes implementar en el algoritmo de multiAgents.py en la clase de AlfhaBhetaAgent para realizar la misma acción de Q2 pero usando la optimización de Alpha Betha para encontrar de manera más efectiva y rápida el árbol de movimientos para pacman.

1. Implementar el algoritmo:

- Implementa el algoritmo en la clase AlfhaBhetaAgent.
- El algoritmo debe retornar el mejor camino posible que debe recorrer pacman.

2. Uso extras:

- De ser necesario utiliza funciones que se encuentran dentro del mismo código o de los imports.

Consideraciones:

- **Solo puedes modificar en la clase de AlfhaBhetaAgent.**

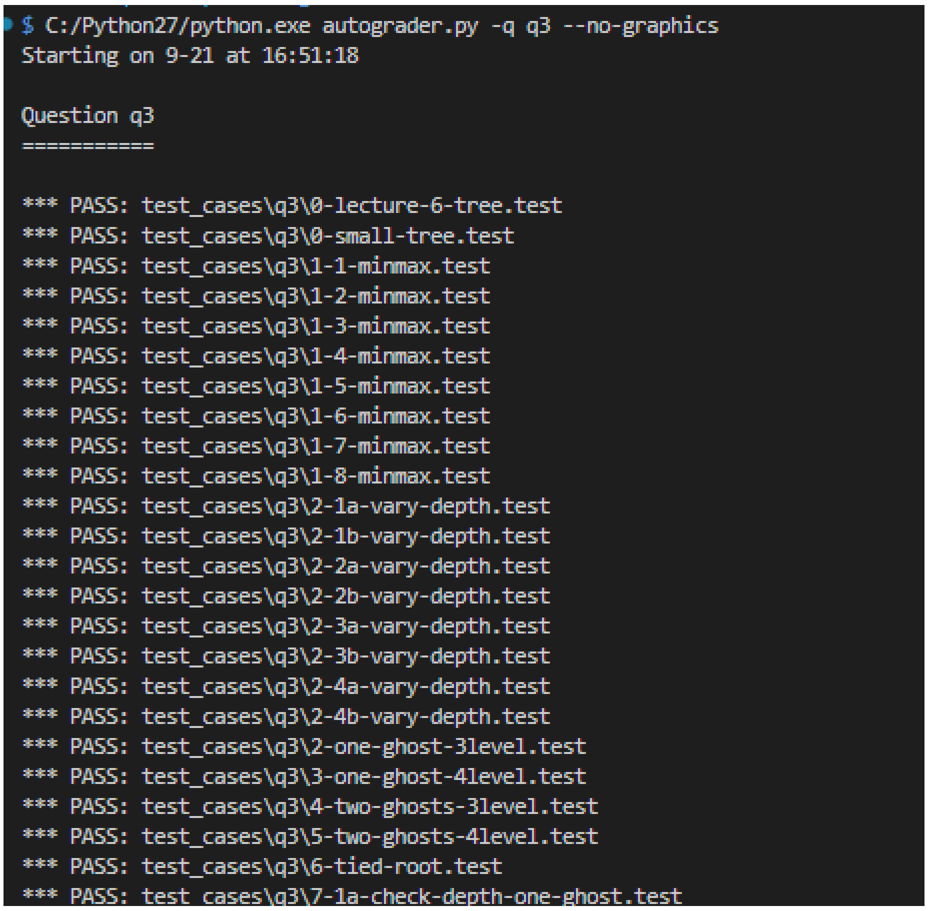
Codigo Inicial:

```
class AlphaBetaAgent(MultiAgentSearchAgent):
  """
    Your minimax agent with alpha-beta pruning (question 3)
  """”
  def getAction(self, gameState):
    """
      Returns the minimax action using self.depth and self.evaluationFunction
    """
    "*** YOUR CODE HERE ***"
    util.raiseNotDefined()
```

Resultado Autograder:

Miguel Angel Nivia Ortega
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

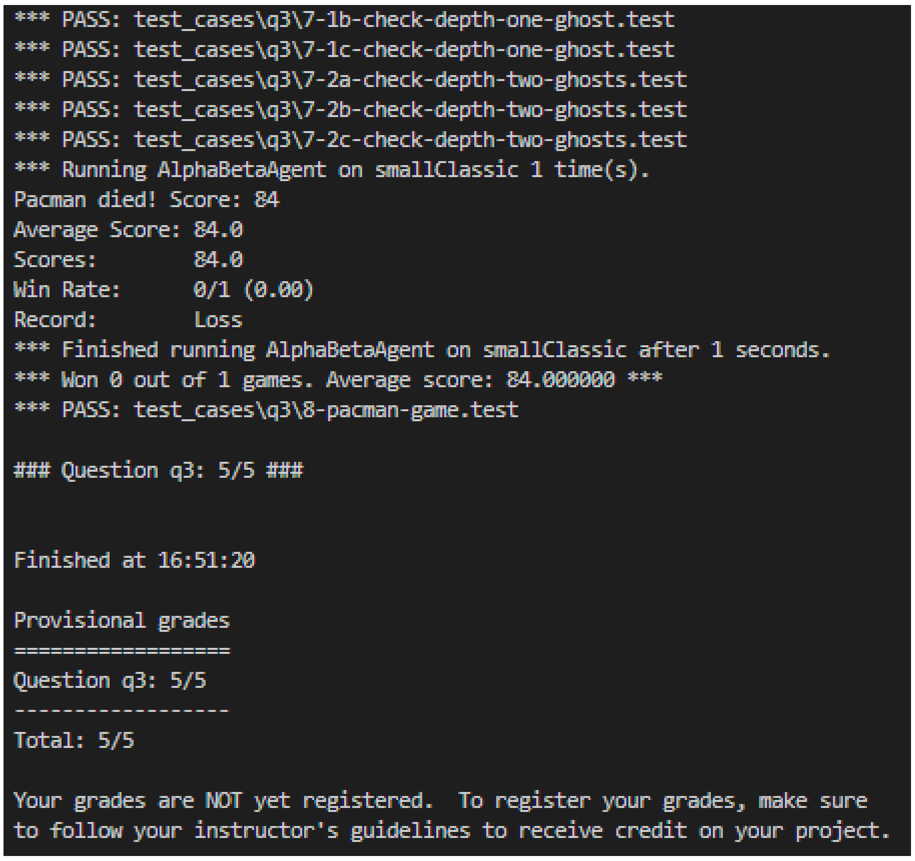
21/09/2024



```
$ C:/Python27/python.exe autograder.py -q q3 --no-graphics
Starting on 9-21 at 16:51:18

Question q3
===========

*** PASS: test_cases\q3\0-lecture-6-tree.test
*** PASS: test_cases\q3\0-small-tree.test
*** PASS: test_cases\q3\1-1-minmax.test
*** PASS: test_cases\q3\1-2-minmax.test
*** PASS: test_cases\q3\1-3-minmax.test
*** PASS: test_cases\q3\1-4-minmax.test
*** PASS: test_cases\q3\1-5-minmax.test
*** PASS: test_cases\q3\1-6-minmax.test
*** PASS: test_cases\q3\1-7-minmax.test
*** PASS: test_cases\q3\1-8-minmax.test
*** PASS: test_cases\q3\2-1a-vary-depth.test
*** PASS: test_cases\q3\2-1b-vary-depth.test
*** PASS: test_cases\q3\2-2a-vary-depth.test
*** PASS: test_cases\q3\2-2b-vary-depth.test
*** PASS: test_cases\q3\2-3a-vary-depth.test
*** PASS: test_cases\q3\2-3b-vary-depth.test
*** PASS: test_cases\q3\2-4a-vary-depth.test
*** PASS: test_cases\q3\2-4b-vary-depth.test
*** PASS: test_cases\q3\2-one-ghost-3level.test
*** PASS: test_cases\q3\3-one-ghost-4level.test
*** PASS: test_cases\q3\4-two-ghosts-3level.test
*** PASS: test_cases\q3\5-two-ghosts-4level.test
*** PASS: test_cases\q3\6-tied-root.test
*** PASS: test_cases\q3\7-1a-check-depth-one-ghost.test
```



```
*** PASS: test_cases\q3\7-1b-check-depth-one-ghost.test
*** PASS: test_cases\q3\7-1c-check-depth-one-ghost.test
*** PASS: test_cases\q3\7-2a-check-depth-two-ghosts.test
*** PASS: test_cases\q3\7-2b-check-depth-two-ghosts.test
*** PASS: test_cases\q3\7-2c-check-depth-two-ghosts.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 1 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q3\8-pacman-game.test

### Question q3: 5/5 ###


Finished at 16:51:20

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

Miguel Angel Nivia Ortega
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

21/09/2024

Q4:

Instrucciones

Debes implementar en el algoritmo de multiAgents.py en la clase de ExpectimaxAgent la realización de un recorte que use el árbol ya creado anteriormente para evitar caminos o opciones ineficientes, que logren llevar pacman a realizar el objetivo de forma adecuada y específicamente para que este detecte a los fantasmas cuando lo pueden atrapar, para evitar quedar encerrado y morir.

1. Implementar el algoritmo:

- Implementa el algoritmo en la clase ExpectimaxAgent.
- El algoritmo debe retornar el mejor camino posible que debe recorrer pacman.

2. Uso extras:

- De ser necesario utiliza funciones que se encuentran dentro del mismo código o de los imports.
- Ademas puedes usar números flotantes para facilitar cálculos.

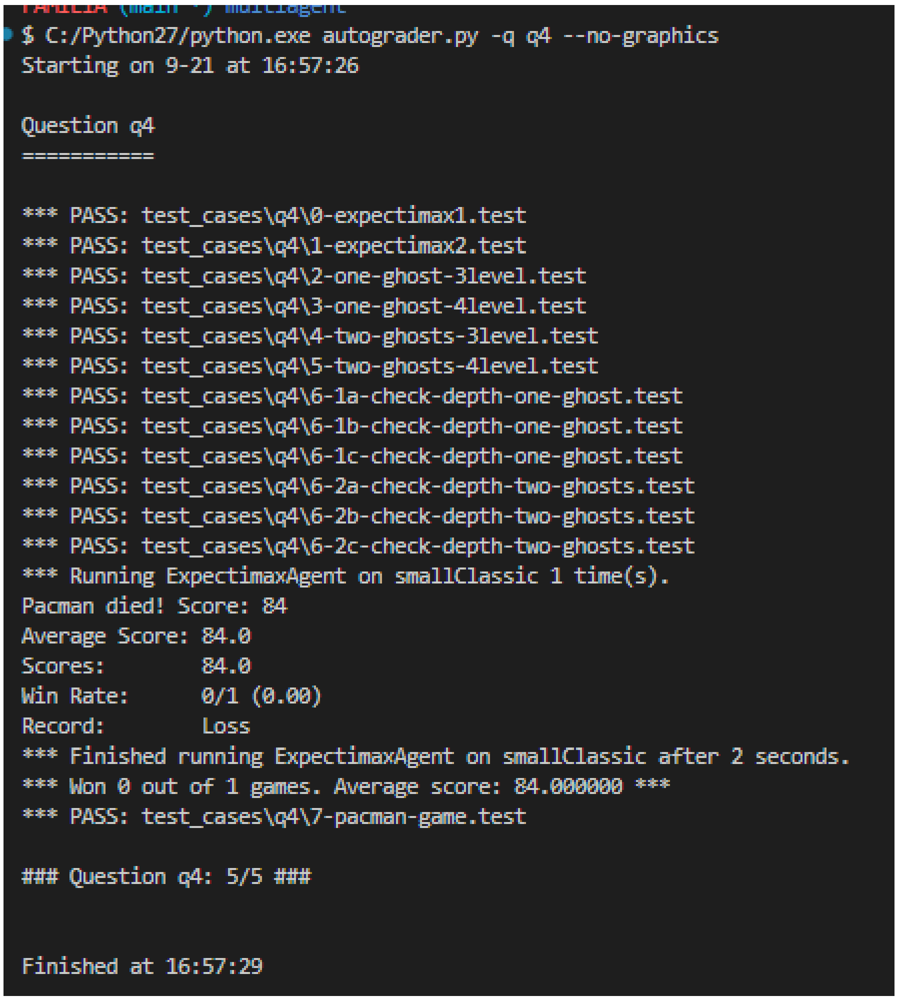
Consideraciones:

- **Solo puedes modificar en la clase de ExpectimaxAgent.**

Codigo Inicial:

```
class ExpectimaxAgent(MultiAgentSearchAgent):
  """
   Your expectimax agent (question 4)
  """
  def getAction(self, gameState):
    """
     Returns the expectimax action using self.depth and self.evaluationFunction
     All ghosts should be modeled as choosing uniformly at random from their
     legal moves.
    """
    "*** YOUR CODE HERE ***"
    util.raiseNotDefined()
```

Resultado Autograder:



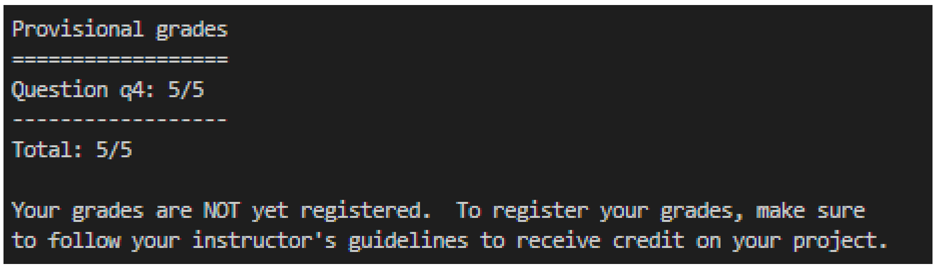
```
$ C:/Python27/python.exe autograder.py -q q4 --no-graphics
Starting on 9-21 at 16:57:26

Question q4
===========

*** PASS: test_cases\q4\0-expectimax1.test
*** PASS: test_cases\q4\1-expectimax2.test
*** PASS: test_cases\q4\2-one-ghost-3level.test
*** PASS: test_cases\q4\3-one-ghost-4level.test
*** PASS: test_cases\q4\4-two-ghosts-3level.test
*** PASS: test_cases\q4\5-two-ghosts-4level.test
*** PASS: test_cases\q4\6-1a-check-depth-one-ghost.test
*** PASS: test_cases\q4\6-1b-check-depth-one-ghost.test
*** PASS: test_cases\q4\6-1c-check-depth-one-ghost.test
*** PASS: test_cases\q4\6-2a-check-depth-two-ghosts.test
*** PASS: test_cases\q4\6-2b-check-depth-two-ghosts.test
*** PASS: test_cases\q4\6-2c-check-depth-two-ghosts.test
*** Running ExpectimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running ExpectimaxAgent on smallClassic after 2 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q4\7-pacman-game.test

### Question q4: 5/5 ###


Finished at 16:57:29
```



```
Provisional grades
==================
Question q4: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

Q5:

Instrucciones

Debes implementar en el algoritmo de multiAgents.py en la función de betterEvaluationFunction donde evalue los ejercicios anteriores y calcule si son correctos con las siguientes instrucciones:

- If you win at least once without timing out the autograder, you receive 1 points. Any agent not satisfying these criteria will receive 0 points.
- +1 for winning at least 5 times, +2 for winning all 10 times
- +1 for an average score of at least 500, +2 for an average score of at least 1000 (including scores on lost games)

Miguel Angel Nivia Ortega **21/09/2024**
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

- +1 if your games take on average less than 30 seconds on the autograder machine. The autograder is run on EC2, so this machine will have a fair amount of resources, but your personal computer could be far less performant (netbooks) or far more performant (gaming rigs).
- The additional points for average score and computation time will only be awarded if you win at least 5 times.

1. Implementar el algoritmo:

- Implementa el algoritmo en la clase betterEvaluationFunction.
- El algoritmo debe retornar la evaluación de cada ejecución posible que debe recorrer pacman.

2. Uso extras:

- De ser necesario utiliza funciones que se encuentran dentro del mismo código o de los imports.

Consideraciones:

- **Solo puedes modificar en la clase de betterEvaluationFunction.**

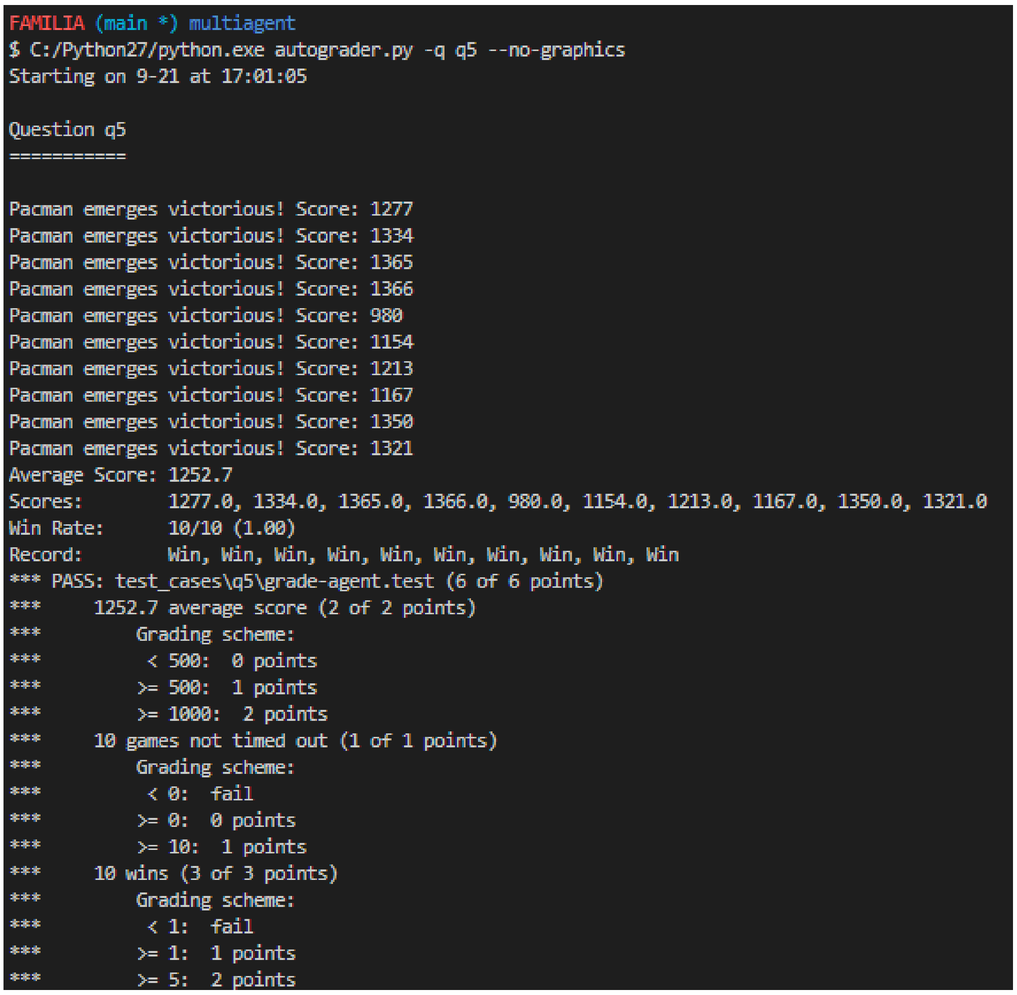
Codigo Inicial:

```
def betterEvaluationFunction(currentGameState):
  """
   Your extreme ghost-hunting, pellet-nabbing, food-gobbling, unstoppable
   evaluation function (question 5).
   DESCRIPTION: <write something here so we know what you did>
  """
  "*** YOUR CODE HERE ***"
  util.raiseNotDefined()
```

Resultados Autograder:

Miguel Angel Nivia Ortega
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

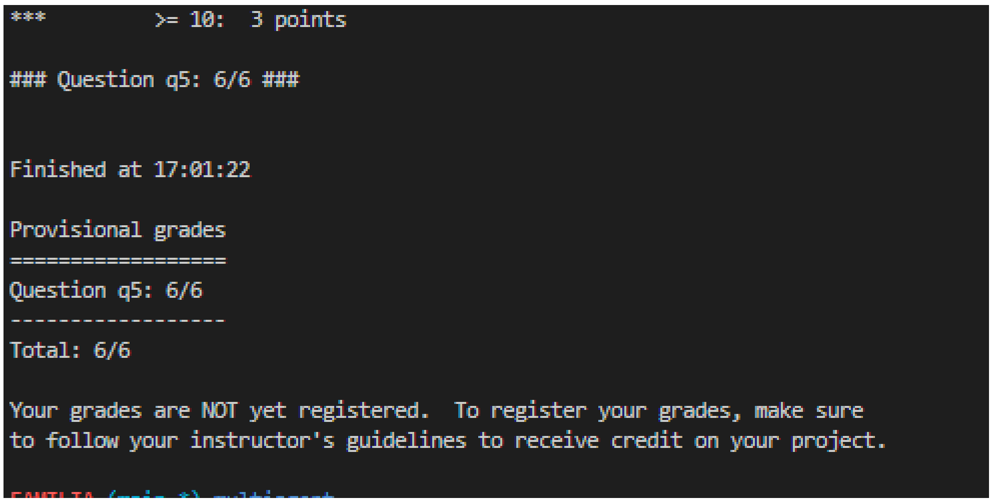
21/09/2024



```
FAMILIA (main *) multiagent
$ C:/Python27/python.exe autograder.py -q q5 --no-graphics
Starting on 9-21 at 17:01:05

Question q5
===========

Pacman emerges victorious! Score: 1277
Pacman emerges victorious! Score: 1334
Pacman emerges victorious! Score: 1365
Pacman emerges victorious! Score: 1366
Pacman emerges victorious! Score: 980
Pacman emerges victorious! Score: 1154
Pacman emerges victorious! Score: 1213
Pacman emerges victorious! Score: 1167
Pacman emerges victorious! Score: 1350
Pacman emerges victorious! Score: 1321
Average Score: 1252.7
Scores:        1277.0, 1334.0, 1365.0, 1366.0, 980.0, 1154.0, 1213.0, 1167.0, 1350.0, 1321.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases\q5\grade-agent.test (6 of 6 points)
***     1252.7 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     10 wins (3 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
```



```
***         >= 10:  3 points

### Question q5: 6/6 ###


Finished at 17:01:22

Provisional grades
==================
Question q5: 6/6
------------------
Total: 6/6

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

FAMILIA (main *) multiagent
```