COEN 266 Artificial Intelligence

Homework #7

The code for this assignment consists of several Python files, some of which you will need to read and understand in order to complete the assignment, and some of which you can ignore. You can download and unzip all the code and supporting files from multiagent.zip.

Submission:

1. Submit a pdf file to Camino (for the format of the file, please refer to Homework7_sample.pdf).
2. Submit all source code needed (with multiAgents.py modified by you) to generate all results of the **Experiments** as a .zip file to Camino. We will test run your submitted code with the three commands in **Experiment 1** and **Experiment 2**, so make sure it works.

Grading: The grade depends on both submitted code and pdf file.

Academic Dishonesty: We will be checking your code against other submissions in the class for logical redundancy. If you copy someone else's code and submit it with minor changes, we will know, and we will pursue the strongest consequences available to us.

Introduction

In this assignment, you will design agents for the classic version of Pacman, including ghosts. The code base has not changed much from the previous Pacman assignment, but please start with the attached multiagent.zip, rather than intermingling files from the previous Pacman assignment.

Files you'll edit:	
multiAgents.py	Where all of your multi-agent search agents will reside.
Files you might want to look at:	
pacman.py	The main file that runs Pacman games. This file also describes a Pacman GameState type, which you will use extensively in this assignment.

game.py	The logic behind how the Pacman world works. This file describes several supporting types like AgentState, Agent, Direction, and Grid.
util.py	Useful data structures for implementing search algorithms. You don't need to use these for this assignment, but may find other functions defined here to be useful.
Supporting files you can ignore:	
graphicsDisplay.py	Graphics for Pacman
graphicsUtils.py	Support for Pacman graphics
textDisplay.py	ASCII graphics for Pacman
ghostAgents.py	Agents to control ghosts
keyboardAgents.py	Keyboard interfaces to control Pacman
layout.py	Code for reading layout files and storing their contents

Welcome to Multi-Agent Pacman

First, play a game of classic Pacman by running the following command:

```
python pacman.py
```

and using the arrow keys to move. Now, run the provided `ReflexAgent` in `multiAgents.py`

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

Note that it plays quite poorly even on simple layouts:

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

Inspect its code (in `multiAgents.py`) and make sure you understand what it's doing.

Task: Reflex Agent

Improve the `ReflexAgent` in `multiAgents.py` to play respectably. The provided reflex agent code provides some helpful examples of methods that query the `GameState` for information. A capable reflex agent will have to consider both food locations and ghost locations to perform well. Your agent should easily and reliably clear the `testClassic` layout:

Experiment 1:

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

Experiment 2:

Try out your reflex agent on the default `mediumClassic` layout with one ghost or two (and animation off to speed up the display):

```
python pacman.py --frameTime 0 -p ReflexAgent -k 1
python pacman.py --frameTime 0 -p ReflexAgent -k 2
```

How does your agent fare? It will likely often die with 2 ghosts on the default board, unless your evaluation function is quite good.