RISK AI Project

This assignment involves designing an AI to play the game of RISK. You should first download the `RISK-AI.zip` file unzip in on your computer. The provided code has been tested with Python 2.7. It might work with changes on other version of Python, but you are on your own for that.

Required: You will need to make sure the Python Imaging Library (PIL) is installed on your system. This can be found at
`http://www.pythonware.com/products/pil/`

The folder you are provided contains:

- `RISK_AI_Handout.pdf` - This file that you are reading
- `README` - Gives instructions on how to run the various tools and programs.
- `matches` - This is the folder where matches that your AI participates in are stored.
- `ai` - This is the folder where the AI files are stored. It is provided with the following baseline AI files for testing and debugging.
  - `random_ai.py` - This is an agent that gets all of the available actions and picks one at random.
  - `attacker_ai.py` - This is an agent that places all troops on territories bordering the opponents and always attacks if that is an option.
  - `donothing_ai.py` - This is an agent that acts randomly, never attacks. A good opponent for early testing.
  - `heuristic_ai.py` - This is an agent that evaluates each possible action using a heuristic function. Currently the heuristic is not implemented, and always returns 0. One possible way to make an AI to complete this assignment is to implement this heuristic function. If you do choose to do this, make sure that it will return different numbers for states that differ only by the result of one action. Sometimes this difference is minimal (one more/less troop in one territory).
- `world.zip` - This file contains all of the map, territory and other files necessary to construct the board and display the game properly.
- `risk.pyw` - This is the interactive game program. Once you run it, you create players, which can be AI's or human (you or your friends). Then start a game. Everything is done by clicking on the map. This will allow you to play against AIs in the AI folder. Games played using this program

are not saved out to a logfile and thus cannot be replayed and watched. To run this program, type:

```
python risk.pyw
```

from the command line. Use your mouse first to add players (choosing an AI for a player if desired) and then using the drop-down menu to start a game. The rest of the game proceeds automatically, with the player input being done by clicking on a territory for placement, attacking, occupation, and fortification.

- `play_risk_ai.py` - This is a script that will automatically perform a match of specified length between the specified AI's and optionally save the game log file. To run the code you should type:

```
python play_risk_ai.py [-h] [-n, --num NUM] [-w, --write] [-v, --verbose]
                       ais
```

  Positional arguments:

  - **ais** List of the AIs for match: ai_1 name_1 ai_2 name_2 . . .

  Optional arguments:

  - **-h, –help** show help message and exit
  - **-n, –num NUM** Specify the number of games each player goes first in match. Default is 5.
  - **-w, –write** Indicate that logfiles should be saved to the matches directory
  - **-v, –verbose** Indicate that the match should be run in verbose mode. This will print out information about each state and action that is visited and can be helpful for debugging.

  Every ai_name should be different, but the same AI file can be submitted for multiple players (with different names.) This function will automatically play all the games and report the results. It will rotate the players order between the different games.

- `risk_game_viewer.py` - This is the script that allows you to view games played using the `play_risk_ai.py` script. It just needs to be passed in the full path to the logfile. Run it by typing:

```
python risk_game_viewer.py LOG_FILE
```

  from the command line.

- `gui` - This is a folder that contains code that supports the interactive gui. You shouldn't need to look at this or modify it.

- `risktools.py` - This files provides the support for the game, and shouldn't be edited. It contains all of the datastructures your AI might need to reference and use. You need to look through it to see what things you can use, but do not edit it or things could stop functioning. Documentation of this module is provided in the next described file.
- `risktools.m.html` - This is documentation for the `risktools.py` module. It will make it easy to see what classes and functions are available for use by your AI.

Rules

Your agent will be playing one-on-one (two-player) RISK and three-player RISK following the rules of RISK as outlined in the rules at

`http://www.hasbro.com/common/instruct/risk.pdf`

(Ignore the special 2-player rules they propose, with buffer armies. We will just play a 2 player game, but with the normal world domination rules.)

Your agent will be presented a state of the game and must return a valid action. Each agent will have a **10 minute time-limit per game** in which to make its decisions. Your agent will be passed the time remaining for each turn as well. You are free to use this time as you see fit, but once you run out of time, the game is over and if you exceed your time limit, you lose automatically.

The games will be capped at **5000 actions**. If the game is not over by this point it will be considered a tie between any players that still possess territories. This limit might be changed later if we find that it is too small to allow for games to normally finish.

A match between two agents will consist of a minimum of 10 games being played, with each player going first an equal number of times. The agent with the most wins out of those games will be considered the winner. In case of a tie, sudden-death play of games will be played with a random player going first.

A match between three agents will consist of a minimum of 15 games being played, with each player going first an equal number of times. The agent with the most wins will be considered the winner. In case of a tie, sudden-death play of games will be played with a random player going first.

Part 1 [100 points]

The main goal of this assignment is for you to write a competitive AI for the game of RISK. You will turn in the agent you write (your `.py` file) along with any supporting files (look-up-tables, etc.) so that you can compete in the tournament.

Beating the random agent [20 points]

You will pass of this part of the assignment if your AI can beat the `random_ai.py`, in a two-player ten game match. To demonstrate this you will submit the logfiles of the match in which your agent beats the `random_ai.py`.

Beating the attacking agent [60 points]

You will pass off this part of the assignment if your AI can beat the `attacker_ai.py`, in a two-player ten game match. To demonstrate this you will submit the logfiles of a match in which your agent beats the `attacker_ai.py`.

Beating two agents [20 points]

You will pass off this part of the assignment if your AI can beat every combination of the `attacker_ai.py` and `random_ai.py` agents (i.e. attacker and attacker, attacker and random, random and random), in a 15 game match. To demonstrate this you will submit the logfiles of three matches, one for each combination.

Part 2 [50 points]

This portion of the part of the assignment requires that you provide a written description of how your agent works and why you designed it like that. This should at least 2 pages, and can be more if you have more details to share.

1 Extra credit [50 points]

Your submitted agent will compete against all of the other AIs from the class in a class competition. Extra credit points will be given out for performance in this competition. 10 extra credit points will be awarded for beating the professor's AI. The competition is currently planned to be a knockout tournament-style format, with tournament seeding done by win-percentage against attacker_ai. The specific competition format might change depending on the run-times of the submitted AIs.