National University of Singapore
School of Computing
CS1010S: Programming Methodology
Semester I, 2017/2018

Contest 15.1
The Hungry Games!

Release date: 01 November 2017
Due: 20 November 2017, 23:59

Required Files

- contest15.1.pdf
- contest15.1.zip

IMPORTANT WARNING: Because we provide you with the flexibility in choosing the approach by which you solve the problems in this mission, we require that you submit well commented/annotated code. Describe your approach to the programming questions, and then annotate the blocks of relevant code that implements your idea. If you fail to do so, you risk having marks deducted by your tutor. Don't assume your tutor can read your mind.

The Hungry Games!

Welcome, welcome to the 4th Hungry Games! Now is the time for you to earn glory for your district. Your tributes will now face off against other tributes, and only the very best will be survive.

Happy Hungry Games and may the odds be ever in your favor.

Administrivia

For this mission, we have provided you with a contest simulation to test your AI agent. Please go to **Coursemology** and download the archived file `contest15.1.zip` and extract it somewhere on your computer. The template file `contest15.1-template.py` is included in the zip file.

This is a continuation from Mission 15, which means all the previous rules applies here too. The allowed (and banned) methods are the same as in Mission 15. You may be disqualified if you violate these rules.

Note: You are allowed to submit your solution from Mission 15 for the contest. Participation is not automatic; you will have to upload your code on Coursemology.

Submission Instructions

Please follow the following naming convention when submitting your file.
`<name-on-coursemology>.py` For example:
`leong-wai-kay.py`

Make sure you have included the following line at the top of your py file:

```
from hungry_games_classes import *
```

Please include only your AI class with the `next_action()` function, and exclude the Testing Code in your submission.

Contest Format

The Hungry Games is an elimination tournament, with each group consisting of up to 6 tributes. The tributes from each group will participate in several rounds of simulations, and the winner of the group will progress to the next stage of the contest.

We have introduced a scoring system that will reward specific actions done by a tribute in each round. The winner of a group will be the tribute with the highest total score from all the rounds of simulations. The score of a tribute will be reset when they progress to the next stage of the contest. In the event of a tie, additional rounds will be simulated until a winner is determined.

Scoring System

At the end of the rounds, points will be given to every tribute. Here is how your tributes can gain points:

a. (Number of tributes - 2) * 50 points for being the survivor of the round. (In the event of a tie, the survivors will split the points)

b. 50 points for killing a tribute

c. 20 points for killing a wild animal

d. 10 points for killing an animal

Wild Animals

To make things more interesting, we have introduced a new Animal - the WildAnimal. A WildAnimal typically have higher health compared to normal Animals, and they may even attack tributes that are in their way! There will be no WildAnimals at the start of the contest; they will be released periodically into the arena.

The damage of the WildAnimal is randomized at the start, but will remain the same throughout the round. You can get the damage that a WildAnimal can do by using the `get_damage()` method of a WildAnimal object. In addition, you can query the probability that a WildAnimal will attack by using the `get_attack_probability()` method.

```
>>> type(bear)
WildAnimal

>>> bear.get_damage()
19

>>> bear.get_attack_probability()
0.2     # 20% probability of attack
```

Additional Information

a. The Game Arena will be a 4x4 map, and it may be wrapped (W) or not wrapped (NW).

b. There will be 6 rounds of simulations for the qualifier rounds, and 10 rounds of simulations for the finals.

c. For every round, your Tribute will start with 200 health and 0 hunger. At the end of every time tick, all tributes will gain 1 hunger. A tribute will die when he/she has 0 health, or 100 hunger, whichever comes first. All tributes will start each round with a Knife (min_dmg of 5, max_dmg of 10).

d. At every time tick, the order of the tribute (and other LivingThing) action is randomized. There will be a limit of 1000 time tick for each Round (but you probably won't be able to last that long ;)

e. The game arena will be reset at the start of every round. We have provided the list of Things, their properties, and their spawn/count rate in the tables below.

f. There will always be an Ammo of a random quantity associated with the RangedWeapon in the map. They may not, however, be present in the same Place.

g. You should not eat everything you can eat. Some 'food' actually makes you more hungry!

Remember: You are not allowed to access the state and properties of any objects directly, and you are only allowed to use specified getters (methods that begin with `get_` and those specified in Mission 15) for the objects. You are also not allowed to "see beyond" your current place and interacting with `Place` objects directly. You may, however, store states in your AI class if necessary.

Tables of Information

Map Size		
Round	**Qualifiers**	**Finals**
1	4x4 NW	4x4 NW
2	4x4 NW	4x4 NW
3	4x4 NW	4x4 NW
4	4x4 W	4x4 NW
5	4x4 W	4x4 NW
6	4x4 W	4x4 W
7	-	4x4 W
8	-	4x4 W
9	-	4x4 W
10	-	4x4 W

Number of objects generated		
Object Type	**Qualifiers**	**Finals**
`Animal`	10	15
`WildAnimal`	3 (every 20 ticks)	5 (every 20 ticks)
`Weapon`	10	15
`RangedWeapon`	10	15
`Food`	10	15
`Medicine`	10	15

Attributes		
Food	**get_food_value()**	
`Carrot`	3-5	
`Apple`	3-5	
`Cabbage`	5-6	
`Potato`	7-9	
`Watermelon`	7-9	
Medicine	**get_food_value()**	**get_medicine_value()**
`Panadol`	1-2	3-5
`Aloe Vera`	1-2	3-5
`Healing Herbs`	0	5-6
`Health Potion`	0	7-9
`Wild Mushroom`	-1	7-9

Attributes			
Animals	**Health Range**	**Food Range**	
Chicken	1-5	5-7	
Sheep	5-10	7-10	
Deer	15-20	6-9	
Pig	15-20	10-12	
Cow	20-25	15-18	
WildAnimals	**Health Range**	**Food Range**	**damage()**
Python	15-20	5-6	1-10
Boar	20-25	20-25	5-10
Wolf	25-30	25-30	10-14
Bear	30-35	30-35	15-20
Mutation	38	-5	20-25
Weapon	**min_damage()**	**max_damage()**	
Knife	5	10	
Dagger	10	12	
Mace	12	15	
Axe	15	20	
Sword	10	25	
Machete	15	25	
RangedWeapon	**min_damage()**	**max_damage()**	**Ammo**
Bow	20	25	Arrows
Crossbow	25	30	Bolts
Pistol	30	40	9mm
Rifle	35	45	556mm

Simulation

In the `contest15.1_template.py` file provided, there is some code at the bottom which would allow you to simulate a contest. The code provided is quite similiar to the actual code that would be run for the qualifiers of the contest. You may change the numbers slightly to simulate the conditions for the Finals as well.

Some additional clarification for the code:

a. The line `match.text_simulate_all()` will simulate all the rounds of the match, and print the textual output to the screen.

b. The line `match.gui_simulate_round(roundId)` will simulate the `roundId`-th round, and show the GUI output. Due to limitation in the animation library, only one round can be displayed at any time.

c. `qualifier_map` is a function which generates the simulation config for a qualifier round. The first parameter is the map size, and the second is a boolean indicating if the map is wrapped or not.

d. You may add your friend's tributes by `import`-ing them. Note that a representation of the AI is a tuple. The first element of the tuple should be the class name of the AI, and the second element is a string containing the name that you want the AI to have. For example, if you want to include Ken's AI into your code:

```
# ken.py

class Ken(Tribute):
    def next_action(self):
        # Ken's AI Code
```

You would include it like this:

```
# soedar.py
from ken import *

class Soedar(Tribute):
    def next_action(self):
        # Soedar's AI Code


...

ken = (Ken, "Ken")
soedar = (Soedar, "Me")

tributes = []
tributes.append(ken)
tributes.append(soedar)
```

e. You may modify the contest parameters, and see how your AI perform under different map/item conditions!