CSCI 1133, Fall 2019
Programming Assignment 11
Due: 11:55pm, Wednesday November 20, 2019

Due Date: Submit your solutions to GitHub by 11:55 p.m., Wednesday, November 20th. We will do a pull from this time point. Do not upload anything to Canvas and PLEASE be sure to use proper naming conventions for the file, classes, and functions. We will NOT change anything to run it using our scripts.

Unlike the computer lab exercises, this is not a collaborative assignment. You must design, implement, and test your code on your own without the assistance of anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class (so you are ONLY allowed to reuse examples from the textbook, lectures, or code you and your partner write to solve lab problems). Otherwise obtaining or providing solutions to any homework problem for this class is considered Academic Misconduct. See the syllabus and read section "Academic Dishonesty" for information concerning cheating. Always feel free to ask the instructor or the TAs if you are unsure of something. They will be more than glad to answer any questions that you have. We want you to be successful and learn so give us the chance to help you.

Instructions: This assignment consists of 2 problems, worth a total of 40 points. Solve the problems below by yourself, and put all functions in a single file called `hw11.py`. Use the signatures given for each class and function. We will be calling your functions with our test cases so you must use the information provided. If you have questions, ask!

Because homework files are submitted and tested electronically, the following are very important:

- You follow all naming conventions mentioned in this homework description.
- You submit the correct file, hw11.py, through Github by the due date deadline.
- You follow the example input and output formats shown.
- Regardless of how or where you develop your solutions, your programs should execute using the python3 command on CSELabs computers running the Linux operating system.

Push your work into Github under your own repo. The specific hosting directory should be: repo-<username>/hw11, where you replace <username> with your U of M user name. For instance, if your email address is bondx007@umn.edu, you should push your hw11 to this directory: repo-bondx007/hw11

The following will result in a score reduction equal to a percentage of the total possible points:

- Incorrectly named/submitted source file, functions, or classes (20%)
- Constraints not followed (40%)
- Failure to execute due to syntax errors (30%)

Use the following template for EVERY function **or method** that you write that requires at least 5 lines of code. The template should appear just before the function/method.

```
#=========================================
# Purpose: (What does the function do?)
# Input Parameter(s): (Each parameter by name and what it represents)
# Return Value(s): (What gets returned? Possibilities?)
#=========================================
```

Problem A. (30 *points*) **RPG Fight!**

You are designing a text-based Role-Playing Game (RPG), in which each character can have one of three jobs: Fighter, Thief, or Wizard. You need a way to keep track of the data for each character, and because a lot of the data and functionality is shared between characters, you have decided to use objects and inheritance to accomplish this.

Part 1:

First, you'll need a base `Adventurer` class, which represents a jobless character template for all of the job classes to build off of.

The `Adventurer` class should have the following methods:

- Constructor: The `Adventurer` class constructor should have the following signature:

  ```
  __init__(self, name, level, strength, speed, power)
  ```

  where `name` (the name of the Adventurer) should be a string, and `level`, `strength`, `speed`, and `power` should be integers that represent the Adventurer's ability levels. The constructor should create instance variables to store the value of each of these arguments (`self.name`, `self.level`, `self.strength`, `self.speed`, and `self.power`, respectively), along with two more instance variables:

  - `self.HP`, an integer representing the Adventurer's remaining Hit Points, which should be initialized to the Adventurer's level multiplied by 6.
  - `self.hidden`, a boolean representing whether the Adventurer is currently hiding, which should be initialized to `False`.

- `__repr__(self)`

  You must overload the `repr()` method (similar to the `str()` method, except that it also changes what is output when you pass the object directly to the interpreter) for the `Adventurer` class so that it returns a string of the format:

  ```
  "<Name> - HP: <Current HP>"
  ```

Be sure to match the spacing and format of the above string exactly or you will fail test cases and lose points (note that the angled brackets are not part of the string itself, but indicate where you should substitute in the actual name and hit points of your adventurer). Note that changing `__repr__` automatically changes the behavior of `str()` as well. See the sample output below for an example.

- `attack(self,target)`

  The `attack` method takes as a parameter another `Adventurer` object `target` and does the following:

  - If the target is hidden, then this method should print the string:

    `"<Name> can't see <Target's name>"`

  - Otherwise, this method should the compute the damage the Adventurer does (their strength + 4), and subtract that value from the target's HP. The method should update `target`'s hit points accordingly, and print out a message of the following format:

`"<Name> attacks <Target's name> for <Damage amount> damage"`

Example: (*`italic text`* is printed, **`bold text`** is returned)

```
>>> thog = Adventurer("Thog",3,5,1,2)
>>> repr(thog)
'Thog - HP: 18'
>>> goku = Adventurer("Goku",20,10,7,9001)
>>> goku
Goku - HP: 120
>>> thog.attack(goku)
Thog attacks Goku for 9 damage
>>> str(goku)
'Goku - HP: 111'
>>> goku.HP
111
>>> goku.hidden = True
>>> thog.attack(goku)
Thog can't see Goku
>>> goku.attack(thog)
Goku attacks Thog for 14 damage
>>> print(thog)
Thog - HP: 4
>>> goku.attack(thog)
Goku attacks Thog for 14 damage
>>> thog.HP
-10
```

Part 2:
Construct three additional classes named `Fighter`, `Thief`, and `Wizard` to represent each of the three possible specializations for an Adventurer. Each of these classes should inherit from the `Adventurer` class and extend it by overloading the constructor and attack methods.

Fighter:

- The `Fighter` constructor must take the same parameters as its superclass constructor and pass them into the superclass constructor, and must then alter `self.HP` to be initialized to level * 12 rather than level * 6.
- The overridden `attack` method for `Fighter` is identical to that of `Adventurer` except that a Fighter does (2*strength + 6) damage to enemies that aren't hidden rather than (strength + 4).

Thief:

- The `Thief` constructor must take the same parameters as its superclass constructor and pass them into the superclass constructor, and must then alter `self.HP` to be initialized to level * 8 rather than level * 6, and set `self.hidden` to `True`.
- The overridden `attack` method for `Thief` should call the superclass method of the same name in the case that the Thief is not hidden (that is, when `self.hidden` is `False`). If the Thief is hidden, then instead the following should occur:
    - If both the Thief and the target are hidden, and the Thief's speed is less than the target's speed, then the Thief is unable to find their target and the method should print a message of the format:

      `"<Name> can't see <Target's name>"`

    - Otherwise, the Thief is able to perform a sneak attack. This does damage equal to (speed + level) * 5. It also causes the Thief and their target (if applicable) to no longer be hidden, and should print a message of the format:

      `"<Name> sneak attacks <Target's name> for <Damage amount> damage"`

Wizard:

- The `Wizard` constructor must take the same parameters as its superclass constructor and pass them into the superclass constructor, and must then initialize a new instance variable, `self.fireballs_left`, initially equal to the Wizard's power.
- The overridden `attack` method for `Wizard` should call the superclass method of the same name in the case that the Wizard has no fireballs left (that is, `self.fireballs_left` is 0). If the Wizard does have fireballs left, the Wizard should instead cast a fireball:

- A fireball causes the target to no longer be hidden (if applicable), and does (level * 3) damage to the target. Remember to subtract one from the Wizard's number of fireballs left. Casting a fireball should also print a message of the format:

  `"<Name> casts fireball on <Target's name> for <Damage amt> damage"`

Part 3:
Finally, create a function (not part of any class) called `duel(adv1, adv2)` which takes two `Adventurer` objects `adv1` and `adv2` as parameters and returns `True` if `adv1` would win in a one-on-one fight, or `False` otherwise. Specifically, the function should do the following:

- If at any point `adv1` reaches 0 or fewer HP while `adv2` still has more than 0 HP, `duel` should print out `adv1` and `adv2`, followed by a message of the form:
  `"<adv2's name> wins!"`
  and then return `False`.

- If at any point `adv2` reaches 0 or fewer HP while `adv1` still has more than 0 HP, `duel` should print out `adv1` and `adv2`, followed by a message of the form:
  `"<adv1's name> wins!"`
  and then return `True`.

- If `adv1` and `adv2` somehow reach 0 HP or below simultaneously (because they start the duel like that, or because they're the same object) `duel` should print out `adv1` and `adv2`, followed by the message:
  `"Everyone loses!"`
  and then return `False`.

- Otherwise, the duel should proceed: `adv1` and `adv2` should be printed out to see the current HP counts, then `adv1` should attack `adv2`, and then `adv2` should attack `adv1`. This should repeat until one of the above conditions is met.

- Remember, you must check whether one or both Adventurers is at 0 or fewer HP after EVERY attack, not just every pair of attacks: if `adv2` reaches 0 or fewer HP after `adv1` attacks, then `adv2` should not get a chance to retaliate.

Constraints:
- Your file should only contain class definitions and the `duel` function. No code should be included outside of the classes/function (though, as usual, comments are fine).
- You must name all classes, methods, and instance variables EXACTLY as specified.
- You should not import any modules.

Examples: (*`italic text`* is printed, **`bold text`** is returned)

Example 1:

```
>>> albus = Wizard("Dumbledore",15,4,6,2)
>>> smeagol = Thief("Gollum",12,1,4,1)
>>> bruce = Thief("Batman",10,4,5,1)
>>> duel(albus,smeagol)
Dumbledore - HP: 90
Gollum - HP: 96
Dumbledore casts fireball on Gollum for 45 damage
Gollum attacks Dumbledore for 5 damage
Dumbledore - HP: 85
Gollum - HP: 51
Dumbledore casts fireball on Gollum for 45 damage
Gollum attacks Dumbledore for 5 damage
Dumbledore - HP: 80
Gollum - HP: 6
Dumbledore attacks Gollum for 8 damage
Dumbledore - HP: 80
Gollum - HP: -2
Dumbledore wins!
True
>>> duel(bruce,albus)
Batman - HP: 80
Dumbledore - HP: 80
Batman sneak attacks Dumbledore for 75 damage
Dumbledore attacks Batman for 8 damage
Batman - HP: 72
Dumbledore - HP: 5
Batman attacks Dumbledore for 8 damage
Batman - HP: 72
Dumbledore - HP: -3
Batman wins!
True
```

Example 2:

```
>>> durden = Fighter("Tyler",6,3,2,1)
>>> duel(durden,durden)
Tyler - HP: 72
Tyler - HP: 72
Tyler attacks Tyler for 12 damage
Tyler attacks Tyler for 12 damage
Tyler - HP: 48
Tyler - HP: 48
Tyler attacks Tyler for 12 damage
Tyler attacks Tyler for 12 damage
Tyler - HP: 24
```

```
Tyler - HP: 24
Tyler attacks Tyler for 12 damage
Tyler attacks Tyler for 12 damage
Tyler - HP: 0
Tyler - HP: 0
Everyone loses!
False
```

Example 3:

```
>>> jack = Thief("Jack",7,2,5,1)
>>> will = Thief("Will",8,3,4,2)
>>> elizabeth = Thief("Elizabeth",5,3,3,2)
>>> duel(will,jack)
Will - HP: 64
Jack - HP: 56
Will can't see Jack
Jack sneak attacks Will for 60 damage
Will - HP: 4
Jack - HP: 56
Will attacks Jack for 7 damage
Jack attacks Will for 6 damage
Will - HP: -2
Jack - HP: 49
Jack wins!
False
>>> duel(jack,will)
Jack - HP: 49
Will - HP: -2
Jack wins!
True
>>> jack.hidden
False
>>> elizabeth.hidden
True
>>> duel(jack,elizabeth)
Jack - HP: 49
Elizabeth - HP: 40
Jack can't see Elizabeth
Elizabeth sneak attacks Jack for 40 damage
Jack - HP: 9
Elizabeth - HP: 40
Jack attacks Elizabeth for 6 damage
Elizabeth attacks Jack for 7 damage
Jack - HP: 2
Elizabeth - HP: 34
Jack attacks Elizabeth for 6 damage
```

```
Elizabeth attacks Jack for 7 damage
Jack - HP: -5
Elizabeth - HP: 28
Elizabeth wins!
False
```

Example 4:

```
>>> sean_bean = Fighter("Boromir",5,5,3,1)
>>> orc1 = Fighter("Orc",1,5,1,1)
>>> orc2 = Fighter("Orc",1,5,1,1)
>>> orc3 = Fighter("Orc",1,5,1,1)
>>> orc4 = Fighter("Orc",1,5,1,1)
>>> duel(orc1,sean_bean)
Orc - HP: 12
Boromir - HP: 60
Orc attacks Boromir for 16 damage
Boromir attacks Orc for 16 damage
Orc - HP: -4
Boromir - HP: 44
Boromir wins!
False
>>> duel(orc2,sean_bean)
Orc - HP: 12
Boromir - HP: 44
Orc attacks Boromir for 16 damage
Boromir attacks Orc for 16 damage
Orc - HP: -4
Boromir - HP: 28
Boromir wins!
False
>>> duel(orc3,sean_bean)
Orc - HP: 12
Boromir - HP: 28
Orc attacks Boromir for 16 damage
Boromir attacks Orc for 16 damage
Orc - HP: -4
Boromir - HP: 12
Boromir wins!
False
>>> duel(orc4,sean_bean)
Orc - HP: 12
Boromir - HP: 12
Orc attacks Boromir for 16 damage
Orc - HP: 12
Boromir - HP: -4
Orc wins!
```

True
```
>>> duel(sean_bean,orc1)
Boromir - HP: -4
Orc - HP: -4
Everyone loses!
```
False

Problem B. (10 *points*) **The Ultimate Showdown**

You have decided to implement a tournament feature within your RPG. Since you want the winner to be somewhat unpredictable, you come up with a scheme to keep the playing field as balanced as possible despite widely varying ability levels: for each round, the two Adventurers with the highest remaining HP will be forced to fight each other until one of them is eliminated. This will hopefully keep the weaker characters in the tournament until the stronger characters have worn each other down.

Create a function called `tournament(adv_list)`, which takes in a single parameter, a list of `Adventurer` objects `adv_list`, and pits them against each other until there is a single winner to return. Specifically:

- If `adv_list` is empty, tournament should return `None`.
- If `adv_list` contains exactly one `Adventurer` object, tournament should return that `Adventurer` object.
- Otherwise, tournament should do the following until there is only one `Adventurer` object remaining:
    - Sort `adv_list` by current HP. You may want to create a key function that returns the current HP of a given `Adventurer` object to use with the built-in `sorted()`, or overload the < operator for `Adventurer`, similar to Homework 10.
    - Pass the second highest HP Adventurer and the highest HP Adventurer into the `duel` function (in that order, so that the second highest HP Adventurer gets to strike first). You may assume that no `Adventurer` object is in the tournament list multiple times or entered the tournament list while already at 0 or less HP, so `duel` is guaranteed to have a clear winner.
    - Whichever Adventurer lost the duel is removed from `adv_list`, and the process repeats. Remember to re-sort the list, as the winner of the duel may now have substantially less HP.

Constraints:

- Your file should only contain class definitions and functions. No code should be included outside of the classes/functions (though, as usual, comments are fine).
- You must name all classes, methods, and instance variables EXACTLY as specified.
- You should not import any modules.

Examples: (*`italic text`* is printed, **`bold text`** is returned)

Example 1:

```
>>> print(tournament([]))
None
```

Example 2:

```
>>> mario = Wizard("Mario",8,4,4,2)
```

```
>>> link = Fighter("Link",7,4,2,1)
>>> fox = Thief("Fox",9,2,4,2)
>>> ness = Wizard("Ness",6,1,1,4)
>>> smashls = [mario,link,fox,ness]
>>> winner = tournament(smashls)
Fox - HP: 72
Link - HP: 84
Fox sneak attacks Link for 65 damage
Link attacks Fox for 14 damage
Fox - HP: 58
Link - HP: 19
Fox attacks Link for 6 damage
Link attacks Fox for 14 damage
Fox - HP: 44
Link - HP: 13
Fox attacks Link for 6 damage
Link attacks Fox for 14 damage
Fox - HP: 30
Link - HP: 7
Fox attacks Link for 6 damage
Link attacks Fox for 14 damage
Fox - HP: 16
Link - HP: 1
Fox attacks Link for 6 damage
Fox - HP: 16
Link - HP: -5
Fox wins!
Ness - HP: 36
Mario - HP: 48
Ness casts fireball on Mario for 18 damage
Mario casts fireball on Ness for 24 damage
Ness - HP: 12
Mario - HP: 30
Ness casts fireball on Mario for 18 damage
Mario casts fireball on Ness for 24 damage
Ness - HP: -12
Mario - HP: 12
Mario wins!
Mario - HP: 12
Fox - HP: 16
Mario attacks Fox for 8 damage
Fox attacks Mario for 6 damage
Mario - HP: 6
Fox - HP: 8
Mario attacks Fox for 8 damage
Mario - HP: 6
Fox - HP: 0
Mario wins!
```

```
>>> winner.name
'Mario'
```

Example 3:

```
>>> jaina = Wizard("Jaina",15,1,2,6)
>>> frisk = Fighter("Frisk",1,1,1,1)
>>> madeline = Thief("Madeline",4,1,10,1)
>>> gandalf = Wizard("Gandalf",19,4,3,1)
>>> arthur = Fighter("Arthur",3,4,4,4)
>>> wesley = Thief("Wesley",14,4,5,1)
>>> winner = tournament([jaina,frisk,madeline,gandalf,arthur,wesley])
Wesley - HP: 112
Gandalf - HP: 114
Wesley sneak attacks Gandalf for 95 damage
Gandalf casts fireball on Wesley for 57 damage
Wesley - HP: 55
Gandalf - HP: 19
Wesley attacks Gandalf for 8 damage
Gandalf attacks Wesley for 8 damage
Wesley - HP: 47
Gandalf - HP: 11
Wesley attacks Gandalf for 8 damage
Gandalf attacks Wesley for 8 damage
Wesley - HP: 39
Gandalf - HP: 3
Wesley attacks Gandalf for 8 damage
Wesley - HP: 39
Gandalf - HP: -5
Wesley wins!
Wesley - HP: 39
Jaina - HP: 90
Wesley attacks Jaina for 8 damage
Jaina casts fireball on Wesley for 45 damage
Wesley - HP: -6
Jaina - HP: 82
Jaina wins!
Arthur - HP: 36
Jaina - HP: 82
Arthur attacks Jaina for 14 damage
Jaina casts fireball on Arthur for 45 damage
Arthur - HP: -9
Jaina - HP: 68
Jaina wins!
Madeline - HP: 32
Jaina - HP: 68
Madeline sneak attacks Jaina for 70 damage
```

```
Madeline - HP: 32
Jaina - HP: -2
Madeline wins!
Frisk - HP: 12
Madeline - HP: 32
Frisk attacks Madeline for 8 damage
Madeline attacks Frisk for 5 damage
Frisk - HP: 7
Madeline - HP: 24
Frisk attacks Madeline for 8 damage
Madeline attacks Frisk for 5 damage
Frisk - HP: 2
Madeline - HP: 16
Frisk attacks Madeline for 8 damage
Madeline attacks Frisk for 5 damage
Frisk - HP: -3
Madeline - HP: 8
Madeline wins!
>>> winner.name
'Madeline'
```