CSE 262: Programming Languages

Fall 2016

Homework 5: **Due on October 28th at 9pm on CourseSite.**

READINGS: Learn You a Haskell: Chapter 6. Real World Haskell: Chapter 4.

SUBMISSION REQUIREMENTS: Submit a text file (.hs/.txt) of your code. **Do not submit a screenshot** of it, you will not receive credit if you do. Your assignment must work on the sunlab machines.

NOTES: Argument order of functions must be in the order specified by the question definition.

In this assignment, you will create a Simplistic Pokemon Combat Simulator (SPCS). This assignment builds onto itself, so be careful.

1. We will handle 5 types of Pokemon: Normal, Water, Fire, Grass, Electric. These are case-sensitive. Create a datatype called Type to represent these. You will want to derive Show, Enum and Eq. This will act as an Enumeration.

   (a) (10 points): Create the function isResistant. Calling it with the Type of the attacker and the defender (in that order) returns whether or not the defender resists the attack (damage - 30).

   The resistance rules are as follows: Water resists Water and Fire. Fire resists Grass and Fire. Grass resists Grass, Water and Electric. Electric resists Electric. Normal resists nothing.

   ```
   isResistant Water Fire -> False
   isResistant Fire Water -> True
   ```

   (b) (10 points): Create the function isVulnerable. Calling it with the Type of the attacker and the defender returns whether or not the defender is vulnerable to the attack (damage * 2).

   The vulnerability rules are as follows: Water is vulnerable to Grass and Electric. Fire is vulnerable to Water. Grass is vulnerable to Fire.

```
isVulnerable Water Fire -> True
isVulnerable Fire Water -> False
```

2. (10 points): Construct the Pokemon type that permits the following Pokemon to be added to your submission. For your purposes, you can, and should, use more than just these. The first value numeric value is how much damage the Pokemon can take (life or hit points). The second numeric value is how much damage the Pokemon does. You will need to derive Show.

```
pikachu = Pokemon "Pikachu" 50 15 Electric
squirtle = Pokemon "Squirtle" 50 20 Water
wartortle = Pokemon "Wartortle" 80 40 Water
snivy = Pokemon "Snivy" 60 20 Grass
tepig = Pokemon "Tepig" 70 30 Fire
jigglypuff = Pokemon "Jigglypuff" 60 20 Normal
```

3. (10 points): Construct the Trainer type that permits the following Trainer objects to be added to your submission. Again, you should test this with additional things. You will need to derive Show.

```
group = Trainer "PGroup" [pikachu, snivy, tepig]
turtles = Trainer "Turtles" [squirtle, wartortle, wartortle]
puff = Trainer "Puffs!" [jigglypuff, jigglypuff, jigglypuff]
```

4. (60 points): You then need to create the combat function that will pit two Trainer objects against each other. Let's discuss the rules.
   (a) Trainer objects will alternate turns with the first Trainer argument taking the first turn.
   (b) The Trainer's first Pokemon (head of the list) will attack the other Trainer's first Pokemon.
   (c) If the defending Pokemon is unable to take any more damage, it is removed.
   (d) If, at any point, a Trainer has no Pokemon left, they lose.
   (e) If there is a winner, return that Trainer's name.
   (f) If, somehow, both Trainers lose, return "No Winner".

(g) If each Trainer has attacked 50 times and no one has won, return "Combat too long!".

```
combat group turtles -> "PGroup"
combat group puff -> "PGroup"
combat turtles group -> "PGroup"
combat turtles puff -> "Turtles"
combat puff turtles -> "Turtles"
combat puff group -> "PGroup"

combat group group -> "Combat too long!"
combat turtles turtles -> "Combat too long!"
combat puff puff ->"Puffs!"
```

(h) You will want some helper and accessory functions.

(i) You will need to keep track of the current "round" of the battle. That would probably be your first helper function.

(j) Decompose the problem. The combat function doesn't need to do any of the battle logic, it just needs to invoke it and check termination conditions.

(k) The battle function doesn't need to remove defeated Pokemon, you can use an accessory function to do that.