Homework #4 **Due: 10/25**

This homework assumes and extends the definitions made in the previous quiz on doing basic nuclear physics with Python, so make sure you have a working implementation of that code first. As before, everything you need to know about the subject is described in this document.

Put all code in a single module named `react` (i.e., a file named `react.py`). It should include a self-test that demonstrates the basic features of the module.

If you attempt any designated extra credit, note that you are doing so in a `README.txt` file included in the tarball.

Remember a guiding principle of software development: "DRY" (Don't Repeat Yourself). Redundant code will lose points.

1. [40 points] Create a class called `Reaction`. This class is created with a left-hand side and a right-hand side, both specified by tuples containing one or more `Particle`s (including `Nucleus`es). The reaction taking place transforms the particles on the left-hand side to the particles on the right hand side.

   Include two exceptions, `UnbalancedCharge` and `UnbalancedNumber` to be invoked as follows. When a reaction is created, your system checks two conservation rules, one for charge and one for mass number:

   - The sums of the charges on the left and right hand sides must be equal. If the charge sums are not equal, your code should "`raise UnbalancedCharge(`*`diff`*`)`," where *diff* is the (`int`) difference in charges.
   - The sums of the mass numbers on the left and right hand sides must be equal. If the mass number sums are not equal, your code should "`raise UnbalancedNumber(`*`diff`*`)`," where *diff* is the (`int`) difference in mass numbers.

   These exceptions should incorporate the *diff*s into their error message *which the exceptions do not print*! Do nothing to handle these exceptions! (Except in your tests.)

   When a reaction is printed, all of the left hand side reactants appear, separated by " `+` ", followed by " `->` ", followed by all of the right hand side reactants, separated by " `+` ".

   Using the particles defined in the quiz:

   ```
   print(Reaction((li6, d), (he4, he4)))
   ```

   should produce

   ```
   (6)Li + (2)H -> (4)He + (4)He
   ```

2. [10 points] Extend the `Particle` class to have the "+" operator acting on two `Particle`s result in a tuple containing them, so that

   ```
   print(Reaction(li6 + d, he4 + he4))
   ```

   is completely equivalent to

   ```
   print(Reaction((li6, d), (he4, he4)))
   ```

   (5 pts. extra credit) Make your code work for more than two `Particle`s being "added", (e.g. "`li6 + d + he4`") and allow particles in `Reaction` arguments to be replicated by replication disguised as positive integer multiplication with the integer on the left (e.g. "`4 * p`" should be treated as "`p + p + p + p`").

3. [50 points] Reactions may be grouped in sets called "chains". Create a `ChainReaction` class that has a name and can contain zero or more `Reactions`. Do this in two parts:

   Create a `ChainReaction` class as specified above. Include a member function `addReaction()` to add a `Reaction` to the chain.

   A chain reaction is then specified as follows:

   ```
   # one of the principal reactions powering the Sun
   he3 = Nucleus("He", 2, 3) # not defined above
   chnPP = ChainReaction("proton-proton (branch I)")
   for rctn in (Reaction((p, p), (d, ep, nu_e)),
                Reaction((p, p), (d, ep, nu_e)),
                Reaction((d, p), (he3, gamma)),
                Reaction((d, p), (he3, gamma)),
                Reaction((he3, he3), (he4, p, p))):
       chnPP.addReaction(rctn)
   ```

   Also, define how a `ChainReaction` is printed. Its format includes its name, each of the constituent reactions, and the "net reaction", so that in the above, "`print(chnPP)`" produces

   ```
   proton-proton (branch I) chain:
   p + p -> (2)H + e+ + nu_e
   p + p -> (2)H + e+ + nu_e
   (2)H + p -> (3)He + gamma
   (2)H + p -> (3)He + gamma
   (3)He + (3)He -> (4)He + p + p
   net:
   p + p + p + p -> e+ + nu_e + e+ + nu_e + gamma + gamma + (4)He
   ```

The net reaction is constructed by the following pseudocode:

```
$lhsNet \leftarrow$ merge of the left-hand sides of all reactions in the chain
$rhsNet \leftarrow$ merge of the right-hand sides of all reactions in the chain
for each particle $p$ in $lhsNet$,
    if $p$ occurs in $rhsNet$,
        remove $p$ from both $lhsNet$ and $rhsNet$
create the net reaction $lhsNet \rightarrow rhsNet$
```

(5 pts. extra credit) Enhance how `Reaction`s and `ChainReaction`s are printed to replace duplicate entries on left- and right-hand sides with counts, so that the above result for the net reaction would be, for instance:

```
4 p -> 2 e+ + 2 nu_e + 2 gamma + (4)He
```

(The order on each side doesn't matter.)