nuSwitch Energy Comparison

Choosing an energy supplier is a difficult job. There are many suppliers with different plans and prices that vary by how much energy a customer consumes.

Your task is to help make the customer's decision easier by writing a program that:

- Prices plans on the market according to how much energy is consumed
- Calculates how much energy is used according to how much a customer spends each month on a specific plan

This document outlines the input your program will receive and the expected output. It also includes a little background information on the problem domain.

Input

There are two inputs to the program: reference data of the plans available is provided in a file called `plans.json`, and a set of command inputs (to be received on `stdin`) is contained in `inputs`.

Plans

Sample `plans.json`:

```
[
  {
    "supplier":         "sse",
    "plan":             "standard",
    "rates":            [
      {"price": 13.5, "threshold": 150},
      {"price": 11.1, "threshold": 100},
      {"price": 10}
    ],
    "standing_charge": 9
  }
  ...
]
```

Plans contain a set of rates that describe how much the customer will be charged for each kilowatt-hour (kWh) of energy that they use. Additionally, plans may also include a daily standing charge.

Plans may have more than one rate but $n-1$ rates will contain a threshold value. Rates are ordered and the last rate will always have no threshold. Thresholds indicate the quantity of energy (up to and including) that may be consumed at that price during the course of the year. Rates without a threshold have no limit.

In the example above, the first 150kWh will be charged at 13.5p/kWh, the next 100kWh will be charged at 11.1p/kWh and all subsequent consumption will be charged at 10p/kWh.

Note that:

- Prices are stated in pence and are *exclusive* of VAT.
- Standing charge is a daily charge stated in pence exclusive of VAT and is applied regardless of consumption.
- VAT for Energy is rated at 5%.

Commands

Example command lines sent on `stdin`:

```
price 1000
price 1200
usage eon variable 40
usage edf fixed 59
exit
```

There are three different commands that will be passed to your program:

- `price ANNUAL_USAGE`

  For a given *annual* kWh consumption produces an annual inclusive of VAT price for all plans available on the market sorted by cheapest first and prints to `stdout`. Each plan will be printed on its own line in the format `SUPPLIER,PLAN,TOTAL_COST`. Total cost should be rounded to 2 decimal places, i.e. pounds and pence.

- `usage SUPPLIER_NAME PLAN_NAME SPEND`

  For the specified plan from a supplier calculates how much energy (in kWh) would be used annually from a monthly spend in pounds (*inclusive* of VAT) rounded to the nearest kWh and prints this value to `stdout`

- `exit` Leaves the program.

Note that all rounding should be natural (i.e. 1.045 rounded to 2 decimal places is 1.05).

Output

Your program will be verified with the provided `inputs` and `plans.json`, and will be expected to produce the exact output specified in `expected_output` (checked using `diff`).

If your program produces `expected_output` it is considered correct.

Your solution

Submitting

Your code should be submitted in a tarball that contains a `./bin/comparison` executable script. This executable will receive the input commands on `stdin` and will be passed a single argument that is the full path to `plans.json`. For example:

```
$ ./bin/comparison /path/to/plans.json < inputs
```

This command will be expected to produce the exact output specified in `expected_output`. It obviously shouldn't just print out the contents of `expected_output` :) We may run your submission against inputs other than those specified/provided but these will be in the same format.

`./bin/comparison` for a Clojure submission may look like this:

```
#!/usr/bin/env bash
lein trampoline run "$@"
```

Your program will be expected to run on OSX or Linux. If your program requires special pre-requisites (if it needs the Go compiler for example) please include it in your submission's README.

Tips

Please write your solution in a language you feel confident in. Your program should both produce the expected output and be well written.

During the in-office interviews we will pair with you to extend the problem (and your program). It may be worth re-reading your code before you come in if you've not looked at it for a while.

Please take your time to solve the problem fully, we appreciate people have day jobs and other commitments. It shouldn't take more than a few hours but please let us know if you need to spread that time over a few weeks.

Please do not publish your solution, for example on your blog or source control site.