Uswitch Extension Exercise

So far, you have priced a collection of plans given a kWh consumption, and also calculated usage given a customer's annual spend.

In this extension exercise, we will be adding a new feature to the program that will apply discounts to your calculation.

New Data

We have provided a new set of `plans`, `inputs` and `expected_output`.

This new data contains the additional attribute; `discounts`.

You should download these files from

http://bit.do/rvu-extension-plan-data

http://bit.do/pricing-plans-with-discounts

http://bit.do/usage-calculation-with-discounts

Types of discount

A plan can now include a discount, or it can have no discount at all.

Discounts are applied to the whole bill, and looks like the following:

```
[
    {
        "applies_to": "whole_bill",
        "amount_in_pence": 500
    }
]
```

- `applies_to` refers to the type of discount being described.

  In this case, `whole_bill` which are lump monetary amounts taken off the bill, in pence.

- `amount_in_pence` is the integer amount to be discounted from the bill

Your task is to update your program to apply the discount, first for the `price`, and then for the `usage` commands.

Notes

- All discounts should be removed **before** VAT is applied
- Discounts are in an **Array**
- There can be either, **zero** or **one** discounts in the array

Testing your changes

To help check the output of your program, we have provided some **Feature files**.

These are natural language descriptions of what your program should do, given certain parameters.

Preparing your Automated Scenarios

Save the discounted plans at the root of your project with the name “plans-with-discounts.json”.

You will need to have the following step definitions that will allow your program to load in the new discounted plans. These can be found in stepdefs.rb/stepdefs.js

```
def load_plans(filepath)
    file = File.open(File.expand_path(filepath))
    data = JSON.load(file.read)
end

Given("the plans provided") do
    @market = EnergyMarket.new(load_plans("../plans.json"))
end

Given("plans with discounts") do
    @market = EnergyMarket.new(load_plans("../plans-with-discounts.json"))
end

When("monthly spend is {float} pounds") do |float|
    @monthly_spend = float
end
```

Running the Automated Scenarios

In the interview you may find it useful to run individual features using tags (e.g add @wip to the top of the feature you are working on)

Ruby

```
cd {coding-test-dir}/ruby
bundle
cucumber
```

You can run an individual scenario using tags

```
cucumber --tags @price
cucumber --tags @usage
```

JS

```
cd {coding-test-dir}/js
npm install
npm test
```

```
{coding-test-dir}/js/node_modules/cucumber/bin/cucumber-js --tags @price
```