Technical evaluation

Objectives:

- Evaluation of programming style
- Evaluation of language knowledge
- Evaluation of testing approach

Technical requirements:

- Use Ruby language (not Ruby on Rails)
- Use TDD (Test-Driven Development) methodology
- Use github as public repository
- Use of DB is not required

Deadline:

- Assignment should be ready within 5 natural days of inbox reception
- Assignment is completed when we receive the link to the repository

Assignment:

You are the lead programmer for a small chain of supermarkets. You are required to make a simple cashier function that adds products to a cart and displays the total price.

You have the following test products registered:

Product code	**Name**	**Price**
GR1	Green tea	£3.11
SR1	Strawberries	£5.00
CF1	Coffee	£11.23

Special conditions:

- The CEO is a big fan of buy-one-get-one-free offers and of green tea. He wants us to add a rule to do this.

- The COO, though, likes low prices and wants people buying strawberries to get a price discount for bulk purchases. If you buy 3 or more strawberries, the price should drop to £4.50

- The CTO is a coffee addict. If you buy 3 or more coffees, the price of all coffees should drop to two thirds of the original price.

Our check-out can scan items in any order, and because the CEO and COO change their minds often, it needs to be flexible regarding our pricing rules.

The interface to our checkout looks like this (shown in ruby):

```
co = Checkout.new(pricing_rules)
co.scan(item)
co.scan(item)
price = co.total
```

Implement a checkout system that fulfills these requirements.

Test data:

Basket: GR1,SR1,GR1,GR1,CF1
Total price expected: **£22.45**

Basket: GR1,GR1
Total price expected: **£3.11**

Basket: SR1,SR1,GR1,SR1
Total price expected: **£16.61**

Basket: GR1,CF1,SR1,CF1,CF1
Total price expected: **£30.57**