Checkout Kata

We're going to see how far we can get in implementing a supermarket checkout that calculates the total price of a number of items. In a normal supermarket, things are identified using Stock Keeping Units, or SKUs. In our store, we'll use individual letters of the alphabet (A, B, C, and so on). Our goods are priced individually. In addition, some items are multipriced: buy n of them, and they'll cost you y pounds. For example, item 'A' might cost 50 pounds individually, but this week we have a special offer: buy three 'A's and they'll cost you 130. The price and offer table:

```
Item  Price   Offer
--------------------------
A     50       3 for 130
B     30       2 for 45
C     20
D     15
```

Our checkout accepts items in any order, so that if we scan+ a B, an A, and another B, we'll recognize the two B's and price them at 45 (for a total price so far of 95).

Getting Started

The interface to your checkout can be anything you like but we'd suggest passing in a string of SKUs to begin with. Here are the first few tests from a Ruby implementation, but feel free to use any language, and write the tests in any order you prefer:

```
def price(goods)
   co = CheckOut.new(RULES)
   goods.split(//).each { |item| co.scan(item) }
   co.total
end

def test_totals
  assert_equal(  0, price(""))
  assert_equal( 50, price("A"))
  assert_equal( 80, price("AB"))
  assert_equal(115, price("CDBA"))
  assert_equal(100, price("AA"))
  assert_equal(130, price("AAA"))
  assert_equal(175, price("AAABB"))
end
```