Coding challenge for itemis

Hi.

Below you will find 3 Problems out of which we ask you to solve exactly one.
We know that the problems below are far from unknown and you will find solutions to them everywhere on github. We encourage you to create your own solution - please don't cheat ;)
To deliver your code please create a git repository on github and send us the link or zip a local git repository and send it to us via mail.
Please focus on code quality and make production ready code. **We ask you to work test driven and make commits so we can understand your process.** You are free to use build systems and libraries the same way you would build real-world software.
Please also add a file that gives us hints on how to build it, and the assumptions that you made.
We would ask you to finish within 7 days, but inform us if you need more time. Once you are confident with your solution, just answer to the mail this file was attached to and give us access to your solution.

Problem 1: SALES TAXES

Basic sales tax is applicable at a rate of 10% on all goods, except books, food, and medical products that are exempt. Import duty is an additional sales tax
applicable on all imported goods at a rate of 5%, with no exemptions. When I purchase items I receive a receipt which lists the name of all the items and their price (including tax), finishing with the total cost of the items,
and the total amounts of sales taxes paid. The rounding rules for sales tax are that for a tax rate of n%, a shelf price of p contains (np/100 rounded up to the nearest 0.05) amount of sales tax.

Write an application that prints out the receipt details for these shopping baskets…

INPUT:
Input 1:
> 1 book at 12.49
> 1 music CD at 14.99
> 1 chocolate bar at 0.85

Input 2:
> 1 imported box of chocolates at 10.00
> 1 imported bottle of perfume at 47.50

Input 3:
> 1 imported bottle of perfume at 27.99
> 1 bottle of perfume at 18.99
> 1 packet of headache pills at 9.75
> 1 imported box of chocolates at 11.25

OUTPUT
Output 1:

> 1 book: 12.49
> 1 music CD: 16.49
> 1 chocolate bar: 0.85
> Sales Taxes: 1.50
> Total: 29.83

Output 2:
> 1 imported box of chocolates: 10.50
> 1 imported bottle of perfume: 54.65
> Sales Taxes: 7.65
> Total: 65.15

Output 3:
> 1 imported bottle of perfume: 32.19
> 1 bottle of perfume: 20.89
> 1 packet of headache pills: 9.75
> 1 imported box of chocolates: 11.85
> Sales Taxes: 6.70
> Total: 74.68

Problem 2: CONFERENCE TRACK MANAGEMENT

You are planning a big programming conference and have received many proposals which have passed the initial screen process but you're having trouble fitting them into the time constraints of the day -- there are so many possibilities! So you write a program to do it for you.

* The conference has multiple tracks each of which has a morning and afternoon session.
* Each session contains multiple talks.
* Morning sessions begin at 9am and must finish by 12 noon, for lunch.
* Afternoon sessions begin at 1pm and must finish in time for the networking event.
* The networking event can start no earlier than 4:00 and no later than 5:00.
* No talk title has numbers in it.
* All talk lengths are either in minutes (not hours) or lightning (5 minutes).
* Presenters will be very punctual; there needs to be no gap between sessions.

Note that depending on how you choose to complete this problem, your solution may give a different ordering or combination of talks into tracks. This is acceptable; you don't need to exactly duplicate the sample output given here.

Test input:
> Writing Fast Tests Against Enterprise Rails 60min
> Overdoing it in Python 45min
> Lua for the Masses 30min
> Ruby Errors from Mismatched Gem Versions 45min

> Common Ruby Errors 45min
> Rails for Python Developers lightning
> Communicating Over Distance 60min
> Accounting-Driven Development 45min

> Woah 30min
> Sit Down and Write 30min
> Pair Programming vs Noise 45min
> Rails Magic 60min
> Ruby on Rails: Why We Should Move On 60min
> Clojure Ate Scala (on my project) 45min
> Programming in the Boondocks of Seattle 30min
> Ruby vs. Clojure for Back-End Development 30min
> Ruby on Rails Legacy App Maintenance 60min
> A World Without HackerNews 30min
> User Interface CSS in Rails Apps 30min

Test output:
> Track 1:
> 09:00AM Writing Fast Tests Against Enterprise Rails 60min
> 10:00AM Overdoing it in Python 45min
> 10:45AM Lua for the Masses 30min
> 11:15AM Ruby Errors from Mismatched Gem Versions 45min
> 12:00PM Lunch
> 01:00PM Ruby on Rails: Why We Should Move On 60min
> 02:00PM Common Ruby Errors 45min
> 02:45PM Pair Programming vs Noise 45min
> 03:30PM Programming in the Boondocks of Seattle 30min
> 04:00PM Ruby vs. Clojure for Back-End Development 30min
> 04:30PM User Interface CSS in Rails Apps 30min
> 05:00PM Networking Event
>
> Track 2:
> 09:00AM Communicating Over Distance 60min
> 10:00AM Rails Magic 60min
> 11:00AM Woah 30min
> 11:30AM Sit Down and Write 30min
> 12:00PM Lunch
> 01:00PM Accounting-Driven Development 45min
> 01:45PM Clojure Ate Scala (on my project) 45min
> 02:30PM A World Without HackerNews 30min
> 03:00PM Ruby on Rails Legacy App Maintenance 60min
> 04:00PM Rails for Python Developers lightning
> 05:00PM Networking Event

Problem 3: MERCHANT'S GUIDE TO THE GALAXY

You decided to give up on earth after the latest financial collapse left 99.99% of the earth's population with 0.01% of the wealth. Luckily, with the scant sum of money that is left in your account, you are able to afford to rent a spaceship, leave earth, and fly all over the galaxy to sell common metals and dirt (which apparently is worth a lot).
Buying and selling over the galaxy requires you to convert numbers and units, and you decided to write a program to help you.

The numbers used for intergalactic transactions follows similar convention to the roman numerals and you have painstakingly collected the appropriate translation between them.

Roman numerals are based on seven symbols:
Symbol Value
I 1
V 5
X 10
L 50
C 100
D 500
M 1,000

Numbers are formed by combining symbols together and adding the values.
For example, MMVI is 1000 + 1000 + 5 + 1 = 2006.

Generally, symbols are placed in order of value, starting with the largest values. When smaller values precede larger values, the smaller values are subtracted from the larger values, and the result is added to the total.
For example MCMXLIV = 1000 + (1000 - 100) + (50 - 10) + (5 - 1) = 1944.

The symbols "I", "X", "C", and "M" can be repeated three times in succession, but no more. (They may appear four times if the third and fourth are separated by a smaller value, such as XXXIX.) "D", "L", and "V" can never be repeated. "I" can be subtracted from "V" and "X" only. "X" can be subtracted from "L" and "C" only. "C" can be subtracted from "D" and "M" only. "V", "L", and "D" can never be subtracted. Only one small-value symbol may be subtracted from any large-value symbol. A number written in [16]Arabic numerals can be broken into digits. For example, 1903 is composed of 1, 9, 0, and 3. To write the Roman numeral, each of the non-zero digits should be treated separately. Inthe above example, 1,000 = M, 900 = CM, and 3 = III. Therefore, 1903 = MCMIII.
(Source: [Wikipedia](http://en.wikipedia.org/wiki/Roman_numerals))

Input to your program consists of lines of text detailing your notes on the conversion between intergalactic units and roman numerals. You are expected to handle invalid queries appropriately.

Test input:
> glob is I
> prok is V
> pish is X
> tegj is L
> glob glob Silver is 34 Credits
> glob prok Gold is 57800 Credits
> pish pish Iron is 3910 Credits
> how much is pish tegj glob glob ?
> how many Credits is glob prok Silver ?
> how many Credits is glob prok Gold ?
> how many Credits is glob prok Iron ?

> how much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Test Output:
> pish tegj glob glob is 42
> glob prok Silver is 68 Credits
> glob prok Gold is 57800 Credits
> glob prok Iron is 782 Credits
> I have no idea what you are talking about