It was great speaking with you! Per our conversation, here’s the next step in our hiring process

Below are three programming problems. Please read all three descriptions thoroughly then create a program to solve ONE of the problems. If you submit more than one solution, we will review only one. For the solution, we request that you use Java, Ruby, or C#.

System security is very important to us and certain file extensions will be blocked for security purposes, resulting in delays to your application. You should NOT include any executable attachments, including those with .exe or .lib extensions. We need to be able to run and build your code ourselves, so please provide a link to your repository on github or a zipped version of your source code to [jennifer.call@truewhitespace.com](mailto:jennifer.call@truewhitespace.com) and [info@truewhitespace.com](mailto:info@truewhitespace.com).

Please include a brief explanation of your design and assumptions, along with your code, as well as detailed instructions to run your application.

We assess a number of things including the design aspect of your solution and your object oriented programming skills. While these are small problems, we expect you to submit what you believe is production-quality code; code that you’d be able to run, maintain, and evolve. You don’t need to gold plate your solution, however we are looking for something more than a bare-bones algorithm.

We want our hiring process to be fair, and for everyone to start from the same place. To enable this, we request that you do not share or publish these problems.

As a general rule, we allow a week from the date that you receive these instructions to submit your code, but you may request more time from your recruiter if needed. If you have any questions about the code as it relates to your interview process, please contact your recruiter.

Whitespace would like the opportunity to offer you a challenging career with our dynamic team. We wish you luck and look forward to receiving your response.

## 

## INTRODUCTION TO THE PROBLEMS

* There must be a way to supply the application with the input data via text file
* The application must run
* You should provide sufficient evidence that your solution is complete by, as a minimum, indicating that it works correctly against the supplied test data

### PROBLEM ONE: SPEEDY LOOP

The local commuter railroad services a number of towns in Iansville. Because of monetary concerns, all of the tracks are 'one-way.' That is, a route from Irving to Invercargill does not imply the existence of a route from Invercargill to Irving. In fact, even if both of these routes do happen to exist, they are distinct and are not necessarily the same distance!

The purpose of this problem is to help the transportation company provide its customers with information about the routes. In particular, you will compute the distance along a certain route, the number of different routes between two towns, and the shortest route between two towns.

Input: A directed graph where a node represents a town and an edge represents a route between two towns. The weighting of the edge represents the distance between the two towns. A given route will never appear more than once, and for a given route, the starting and ending town will not be the same town.

Output: For test input 1 through 5, if no such route exists, output 'NO SUCH ROUTE'. Otherwise, follow the route as given; do not make any extra stops! For example, the first problem means to start at city A, then travel directly to city B (a distance of 5), then directly to city C (a distance of 4).

1. The distance of the route A-B-C.
2. The distance of the route A-D.
3. The distance of the route A-D-C.
4. The distance of the route A-E-B-C-D.
5. The distance of the route A-E-D.
6. The number of trips starting at C and ending at C with a maximum of 3 stops. In the sample data below, there are two such trips: C-D-C (2 stops). and C-E-B-C (3 stops).
7. The number of trips starting at A and ending at C with exactly 4 stops. In the sample data below, there are three such trips: A to C (via B,C,D); A to C (via D,C,D); and A to C (via D,E,B).
8. The length of the shortest route (in terms of distance to travel) from A to C.
9. The length of the shortest route (in terms of distance to travel) from B to B.
10. The number of different routes from C to C with a distance of less than 30. In the sample data, the trips are: CDC, CEBC, CEBCDC, CDCEBC, CDEBC, CEBCEBC, CEBCEBCEBC.

#### Test Input:

For the test input, the towns are named using the first few letters of the alphabet from A to D. A route between two towns (A to B) with a distance of 5 is represented as AB5.

Graph: AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7

#### Test Output:

Output #1: 9

Output #2: 5

Output #3: 13

Output #4: 22

Output #5: NO SUCH ROUTE

Output #6: 2

Output #7: 3

Output #8: 9

Output #9: 9

Output #10: 7

### PROBLEM TWO: CONFERENCE TRACK MANAGEMENT

You are planning a big software development conference and have received several proposals and 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:

Proper Unit Tests for Anyone **60min**

Why Python? **45min**

TDD for Embedded Systems **30min**

Dependency Management with Interpreted languages **45min**

Java 8, really end of life **45min**

From Java 8 to Java 12 **lightning**

Managing Network Latency **60min**

BDD Gone Mad **45min**

Do you smell that? (Code Smells) **30min**

Open Office Space or Closets???? **30min**

Proper Pairing **45min**

Spring JuJu **60min**

Effective DSL (Domain Specific Languages) **60min**

Clojure... What Happened (on my project) **45min**

Effective Legacy Code Techniques... **30min**

Backends and FrontEnds in JavaScript? **30min**

CD on the Mainframe **60min**

Making Windows Development Enjoyable... **30min**

Modern Build Systems **30min**

#### Test output:

Track 1:

09:00AM Proper Unit Tests for Anyone **60min**

10:00AM Why Python? **45min**

10:45AM TDD for Embedded Systems **30min**

11:15AM Dependency Management with Interpreted languages **45min**

12:00PM Lunch

01:00PM Effective DSL (Domain Specific Languages) **60min**

02:00PM Java 8, really end of life **45min**

02:45PM Proper Pairing **45min**

03:30PM Effective Legacy Code Techniques... **30min**

04:00PM Backends and FrontEnds in JavaScript? **30min**

04:30PM Modern Build Systems **30min**

05:00PM Networking Event

Track 2:

09:00AM Managing Network Latency **60min**

10:00AM Spring JuJu **60min**

11:00AM Do you smell that? (Code Smells) **30min**

11:30AM Open Office Space or Closets???? **30min**

12:00PM Lunch

01:00PM BDD Gone Mad **45min**

01:45PM Clojure... What Happened (on my project) **45min**

02:30PM Making Windows Development Enjoyable... **30min**

03:00PM CD on the Mainframe **60min**

04:00PM From Java 8 to Java 12 **lightning**

05:00PM Networking Event

### PROBLEM THREE: 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 follow a 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

[2]I

[3]1

[4]V

[5]5

[6]X

[7]10

[8]L

[9]50

[10]C

[11]100

[12]D

[13]500

[14]M

[15]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. In the above example, 1,000 = M, 900 = CM, and 3 = III. Therefore, 1903 = MCMIII.

(Source: Wikipedia ( [17]<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