We ask that you read all three descriptions thoroughly then create a program to solve **ONE** of the problems. If you choose to do more than one problem, we will choose and evaluate only one of your solutions.

For the solution, we request that you use either Java, Ruby or C#. You may not use any enhanced functionality of the language or any external libraries to solve this problem; however, you may use external libraries or tools for building or testing purposes. Specifically, you may use JUnit or Ant to assist your development. For security reasons, please do NOT submit your C# code as a .msi file.

You may also include a brief explanation of your design and assumptions along with your code. We expect all code solutions to include a suite of non-trivial unit tests using a testing framework of your choice.

INTRODUCTION TO THE PROBLEMS

All problems below require some kind of input. You are free to implement any mechanism for feeding input into your solution (for example, using hard coded data within a unit test). You should provide sufficient evidence that your solution is complete by, as a minimum, indicating that it works correctly against the supplied test data.

Optional Objectives

Want to impress us? Two of the problems below have additional, optional exercises. You won't be penalized for skipping the optional objectives, but if you have the time free and you want to showcase how you might extend your application, we've provided some starting points below.

PROBLEM ONE: TRAINS

The local commuter railroad services a number of towns in Kiwiland. Because of monetary concerns, all of the tracks are ‘one-way.' That is, a route from Kaitaia to Invercargill does not imply the existence of a route from Invercargill to Kaitaia. 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 railroad 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:
    1. C-D-C (2 stops).
    2. 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:
    1. A to C (via B,C,D);
    2. A to C (via D,C,D);
    3. 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:
    1. CDC,
    2. CEBC,
    3. CEBCDC,
    4. CDCEBC,
    5. CDEBC,
    6. CEBCEBC,
    7. 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

Expected 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
```

Optional Objectives

None. This one seems challenging enough on its own.

PROBLEM TWO: 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 allows a user to add a set of products to a shopping cart, and prints out a receipt listing the individual items, any tax applied, and a total amount paid. At a minimum, your test cases should include the following carts:

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 box of imported 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

Optional Objectives

Want to impress us? Try extending your application with one or more of the following:

- Inventory management - allow a user to manage the list of products available, add, remove, or change the items available in the store.
- Tax-free states - some states don't charge a sales tax; modify your storefront to allow a user to choose their state, and calculate whether tax should be charged based on that state's rules. For testing purposes, let's assume any state that starts with the letter "M" doesn't charge sales tax.

PROBLEM THREE: MARS ROVERS

A squad of robotic rovers are to be landed by NASA on a plateau on Mars. This plateau, which is curiously rectangular, must be navigated by the rovers so that their on-board cameras can get a complete view of the surrounding terrain to send back to Earth.

A rover's position and location is represented by a combination of x and y co-ordinates and a letter representing one of the four cardinal compass points. The plateau is divided up into a grid to simplify navigation. An example position might be 0, 0, N, which means the rover is in the bottom left corner and facing North.

In order to control a rover, NASA sends a simple string of letters. The possible letters are 'L', 'R' and 'M'. 'L' and 'R' makes the rover spin 90 degrees left or right respectively, without moving from its current spot. 'M' means move forward one grid point, and maintain the same heading.

Assume that the square directly North from (x, y) is (x, y+1).

INPUT

The first line of input is the upper-right coordinates of the plateau, the lower-left coordinates are assumed to be 0,0.

The rest of the input is information pertaining to the rovers that have been deployed. Each rover has two lines of input. The first line gives the rover's position, and the second line is a series of instructions telling the rover how to explore the plateau.

The position is made up of two integers and a letter separated by spaces, corresponding to the x and y co-ordinates and the rover's orientation.

Each rover will be finished sequentially, which means that the second rover won't start to move until the first one has finished moving.

OUTPUT

The output for each rover should be its final co-ordinates and heading.

Test Input:
```
5 5
1 2 N
LMLMLMLMM
3 3 E
MMRMMRMRRM
```

Expected Output:
```
1 3 N
5 1 E
```

Optional Objectives

Want to impress us? Try extending your application with one or more of the following:

- Boundary Detection - what happens when you try and send the rover off the grid? Does it stop in place, or careen off into the unknown?
- Garbled Input - if a rover receives an invalid command string, what does it do? Does it execute up to the point of the broken instruction, or does it wait to move until it can verify the entire command string?
- Graphical Representation - everyone loves graphics! Extend your program to print a visual representation of the grid (ASCII is fine) showing the motion of your rover.