


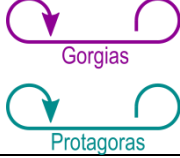
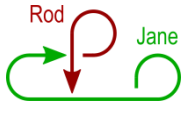
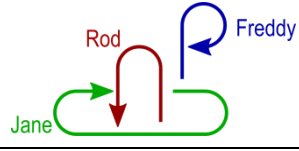
Roboreus Java Programming Test #3.0.7: Snaky Serenades

Background

Your friend Snana keeps special singing snakes as pets. Under the rules of the Singing Snakes Society (SSS), in order for the snakes to sing songs, they are arranged by their keeper into one of many possible ensemble structures. In any such ensemble:

- There may be any number of snakes, each of which has a unique name
- Once arranged, snakes do not move around. (Singing snakes are not slippery.)
- Each snake has its tail-end wrapped around the body of either itself or another snake in the ensemble.
- Each snake has its head-end wrapped around the body of either itself or another snake in the ensemble.
- Snakes sing one at a time; each sings only its own name, and always at the same (unique) pitch (which it is not important to know for the purposes of this problem).
- The songs sung by the snakes are composed by the snake keeper, according to the following rules:
 - Any snake may begin a song (by singing its name).
 - If snake A has its head-end wrapped around snake B, then snake B may sing immediately after snake A
 - If snake C has its tail-end wrapped around snake A, then snake C may sing immediately after snake A
- The songs a particular ensemble may perform are categorised (and limited in length) by the maximum repeats per snake (MRPS) allowed. In a 0-MRPS song, none of the snakes in the ensemble sings *more* than once (but not every snake has to sing); in a 1-MRPS song, no snake in the ensemble may sing more than twice (but no snake *has* to sing as many times as this); and so on. So in an N-MRPS song, each snake sings *at most* N+1 times. Clearly each N-MRPS song is also an (N+1)-MRPS song for the same ensemble.

Snake ensembles can be represented as a type of nodeless hypergraph, where each directed edge corresponds to one of the snakes in the ensemble. Thus ensembles are often drawn as in the diagrams below (the head of each arrow corresponding to the head-end of the snake it represents):

Ensemble				
All 0-MRPS songs	<ul style="list-style-type: none"> • Gorgias 	<ul style="list-style-type: none"> • Gorgias • Protogoras 	<ul style="list-style-type: none"> • Rod • Jane • Rod, Jane • Jane, Rod 	<ul style="list-style-type: none"> • Rod • Jane • Freddy • Jane, Freddy • Jane, Rod • Rod, Jane • Rod, Jane, Freddy
All 1-MRPS songs that are not 0-MRPS songs	<ul style="list-style-type: none"> • Gorgias, Gorgias 	<ul style="list-style-type: none"> • Gorgias, Gorgias • Protogoras, Protogoras 	<ul style="list-style-type: none"> • Rod, Rod • Jane, Jane • Rod, Jane, Jane • Rod, Jane, Rod • Jane, Rod, Rod • Jane, Rod, Jane • Rod, Jane, Jane, Rod • Jane, Rod, Rod, Jane • Jane, Jane, Rod, Rod • Rod, Rod, Jane, Jane • Jane, Rod, Jane, Rod • Jane, Jane, Rod • Rod, Rod, Jane • Rod, Jane, Rod, Jane 	<ul style="list-style-type: none"> • Rod, Jane, Rod • Rod, Jane, Rod, Jane • Jane, Rod, Jane, Freddy • Rod, Jane, Rod, Jane, Freddy • Freddy, Freddy • Jane, Rod, Jane, Freddy, Freddy • Rod, Jane, Rod, Jane, Freddy, Freddy • Rod, Jane, Freddy, Freddy • Jane, Rod, Jane • Jane, Rod, Jane, Rod • Jane, Freddy, Freddy

The SSS stipulates a Standard Snake Ensemble Serialisation File Format (SSESFF) for storing details of snake ensembles. The SSESFF is an ASCII CSV file in which each line contains the following fields:

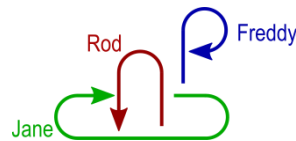
Name, Tail, Head

Where:

- Name is the unique name of the snake
- Tail is the line number in the CSV file of the snake around which this one's tail-end is wrapped
- Head is the line number in the CSV file of the snake around which this one's head-end is wrapped

For the purposes of the “Head” and “Tail” fields, line numbers in the CSV-file are assumed to start from 0. (Note too that SSESFF is not “strict CSV”, in that it not only permits but encourages spaces after commas.)

The SSESFF does not specify the order in which snakes' details are written to the file. Thus the same ensemble can be stored in several different ways. For instance, consider this ensemble:



It could be stored in several ways, including both:

```
Rod, 1, 1
Jane, 0, 0
Freddy, 1, 2
```

and:

```
Jane, 2, 2
Freddy, 0, 1
Rod, 0, 0
```

Task

Your friend Snana is interested in generating N-MRPS songs of maximum length for given ensembles and values of N. She has asked you to write a program to help her.

Your task is to write a Java command-line program for Snana that takes two parameters – the name of an SSESFF file (potentially prefixed by a full or relative directory path) followed by an MRPS number – and outputs (in the same format as the following examples) a maximum-length song for the specified ensemble and MRPS number. So, assuming the main() method of your program is in a class called LongSnakeSong, and that ensemble.csv is a file in the current directory containing any valid SSESFF file for the ensemble in which Rod, Jane and Freddy are arranged above, your program should produce the following output:

```
$ java LongSnakeSong ensemble.csv 0
A maximal length 0-MRPS song for this 3-snake ensemble has 3 sings:
Rod, Jane, Freddy.

$ java LongSnakeSong ensemble.csv 1
A maximal length 1-MRPS song for this 3-snake ensemble has 6 sings:
Rod, Jane, Rod, Jane, Freddy, Freddy.

$
```

Notes

- In the event that there is more than one maximum-length song for a specified ensemble and MRPS number, your program only has to output one of these songs. It does not matter which one.
- Your friend Snana, who is a good Java programmer, needs to be able to understand and maintain and extend the code without reference to you or to any external documentation. (You may assume she is familiar with the SSESFF.) You would like to impress her with how well-commented and production-ready your code is.
- This is your chance to show that you can write idiomatic Java code (as opposed to, say, Java-in-the-style-of-C).