Problem Description

To see all of these concepts in action, we are going to consider a modestly-sized programming example in detail (we note that this is still a rather small program in the grand scheme of things).

Write a program, roster, which reads in a file that has the following format:

```
Number of students
Student Name
Number of classes
Classname 1
Classname 2
...
Classname N
Student Name
```

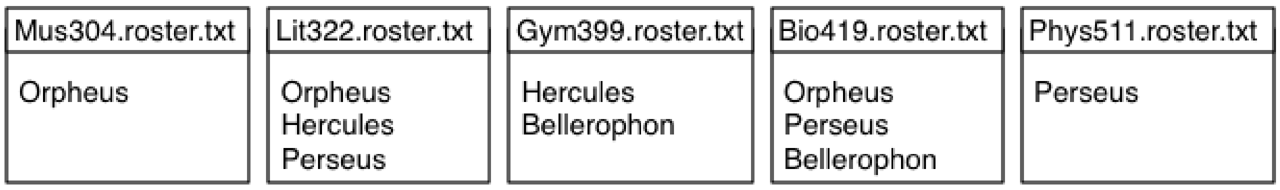
```
Number of classes

Classname 1

Classname 2

...

Classname N
```

That is, the first line contains a single integer, which is the number of students described in the file. After that, there are student descriptions, which each start with a line containing the student's name. The next line is the number of classes that the student is in. After that, there are the specified number of lines, each of which contains the name of one class. Your program should read this file in, and then output the class roster for each class. Specifically, for each class, it should create a file called **classname.roster.txt** (where classname is the name of the class) and print each of the students who are in that course (one per line) into that file.

For example, if your input file were:

```
16

17

18

19

4

Orpheus

3

Mus304

Lit322

Bio419

Hercules

2

Gym399

Lit322

Perseus

3

Lit322

Phys511

Bio419

Bellerophon

2

Gym399

Bio419
```

Then your program would create the following 5 output files:



Note that if the file does not obey this format, your program should print a meaningful error message to stderr and exit with a failure status.

This problem is far too large to write as a single function. Doing this task well involves breaking it down into about a dozen functions. We can intuitively see the need to break things down just by observing that there are several discrete tasks we will need to perform: reading our input (which itself has smaller sub-tasks, such as reading a number with error checking, and reading the information for one student), "rearranging" the information to the format we need, writing the output (which itself has smaller sub-tasks, such as computing the filename for a class and writing out the information for a class), and freeing the memory we allocated when we read the input.

The next video will show how to approach this problem with the Seven Steps.

✓

Completed