Eiffel Project −Simple Recommender

For this assignment you will implement a "recommendation" scheme similar to ones used in e−commerce systems. The input will be a number of ratings for a small group of people, which will then be used to determine whether a movie (in our case) could be recommended to a person. Unlike the project I tortured some of you with in data structures, we will use a very simple form of recommendation.

Sample Input

The following is a sample input file showing the format of the data that your program should read. The file will always be named "input.txt".

```
insert        ABBY
rating        Austin Powers      dislike
rating        Braveheart         like
rating        Castaway           like
rating        Don Juan DeMarco   like
rating        Emma               like
rating        Faceoff            dislike
rating        Goodfellas         dislike
rating        Heathers           dislike
rating        Jane Eyre          like
end
insert        DAPHNE
rating        Austin Powers      like
rating        Braveheart         dislike
rating        Castaway           dislike
rating        Don Juan DeMarco   dislike
rating        Emma               dislike
rating        Jane Eyre          dislike
end
insert        CHARLES
rating        Austin Powers      like
rating        Castaway           like
rating        Don Juan DeMarco   like
rating        Emma               like
rating        Faceoff            dislike
rating        Heathers           like
rating        Il Postino         like
rating        Jane Eyre          like
end
insert        BERNIE
rating        Austin Powers      like
rating        Braveheart         dislike
rating        Castaway           dislike
rating        Emma               dislike
rating        Faceoff            like
rating        Goodfellas         like
rating        Il Postino         dislike
rating        Jane Eyre          dislike
end

query         ABBY               CHARLES      Brave
heart
query         ABBY               CHARLES      Goodf
ellas
query         CHARLES            ABBY         Il
Postino
query         CHARLES            ABBY         Brave
heart
query         DAPHNE             ABBY         Il
Postino
query         BERNIE             ABBY         Il
Postino
```

The file will have two parts. The first part consists of user rating information, and the second part consists of queries that request a predicted rating based on the information that was previously given. The two parts of the file are separated by a blank line, which can be used to determine when to stop reading the rating information.

Each users rating information begins with a line

```
        insert      <name>
```

where `<name>` is a string of non−whitespace characters. The actual ratings are given by one or more lines of the form

```
        rating      <movie>     <preference>
```

where `<movie>` is a string of words possibly separated by spaces, and `<preference>` is either the word `like` or `dislike`. The example shows the entries as sorted, but you should not assume that the data in the input is sorted. A user's ratings end with the line

```
        end
```

All data fields are separated by tab characters.

The queries have the form

```
        query <name>      <name>      <movie>
```

with tabs in between all data fields. This query translates to the question "does the first person recommend the movie to the second person?" For instance, the first query in the example asks "does Abby recommend Braveheart to Charles?"

Queries are to be answered by comparing the ratings of one person with those of another. We will use the rule that someone can make a recommendation for another person if their ratings agree for a simple majority of their commonly rated movies. For instance, Abby and Charles can trade recommendations, because they agree on five of seven ratings. If one person can recommend for another person, then we just give the preference of that first person. So, in answer to the query "does Abby recommend Braveheart to Charles?" the answer should be "yes". If we were to ask if Bernie recommends anything to Abby the answer should be "no".

Output

Your output must be printed to a file name "`report.txt`". The output should consist of two lines identifying you as the programmer and the project, and the answers to the queries. The output for the above input would be printed as follows.

```
Programmer: Ben Keller
CS  3304 Project 3 Spring 2002

Query: Does ABBY recommend Braveheart to CHARLES?
Yes
Query: Does ABBY recommend Goodfellas to CHARLES?
No
Query: Does CHARLES recommend Il Postino to ABBY?
Yes
Query: Does CHARLES recommend Braveheart to ABBY?
ABBY likes Braveheart
Query: Does DAPHNE recommend Il Postino to ABBY?
DAPHNE has not rated Il Postino
Query: Does BERNIE recommend Il Postino to ABBY?
BERNIE cannot recommend for ABBY
```

Take note of the answers to the last three queries which correspond to the cases where the person for which the query is being made has already rated the movie, the person who is supposed to make the recommendation cannot because they have not rated the movie, and the case where the person who is to make the recommendation does not agree sufficiently with the other person.

Be extremely careful when printing your output to make sure that yours matches mine. In particular, make sure that you use punctuation as shown, and that there are no extra white−space characters before punctuation characters. Also be careful with capitalization, and you should not change the case of any string read from input. Your output must have the exact same number of lines, but the spacing on the lines does not have to be exact. You must have a blank line where I have a blank line, and similarly if I have something on a line (even if it ends up not being graded) you must have something on that line.

These sample files will be posted on the web.

Design & Implementation

You are to write your solution in Eiffel (using no external routines). You must write and use multiple classes for this project. There is no specific requirement as to what these classes must be, but you should try to make them as natural to the problem as possible. Your choices should be reasonable. A solution that I wrote to this project in Java included a class for each individual rating (a movie−rating pair), a class for the set of ratings for a person, and a class for the database of ratings plus the main program in another class.

You should attempt to use as many library classes as you can in your implementation. You can find out about the class interfaces by following the Library Interfaces link at the SmallEiffel website. There may be others, but DICTIONARY[Value,Key] and SET[E] look particularly appropriate. You can also find more about the STRING and other standard classes there. There are some discrepancies between the library interface and the actual files, so before getting caught up in using a class you might want to check that it exists in your local installation. On the installation on my Mandrake Linux machine these files are in /usr/lib/SmallEiffel/lib_std. The machines in 124 should have the files in the same place. You may also find the examples in /usr/lib/SmallEiffel/lib_show a useful reference (although there don't appear to be any that do file I/O). The ACE examples might be helpful in setting up the build for your system.

For input look at STD_FILE_READ (which is a class and not an interface as shown on the website). Some useful capabilities for your purposes are defined in INPUT_STREAM from which the STD_FILE_READ class inherits. For output STD_FILE_WRITE and OUTPUT_STREAM.

Your classes should be encapsulated to hide any unnecessary implementation details. You may use feature access control to publish different interfaces for your classes. Style should follow the examples in the library with appropriate block comments in each file that indicate that you are the author. There is an emacs mode available on the SmallEiffel website at the bottom of the Repository page.

Submitting your Program and Evaluation

You will submit this assignment to the Curator System (read the *Student Guide*) as a zip file. Your submission should only include your Eiffel and any files needed to build the program, and a README explaining how to build the program. If your program doesn't work or is incomplete, include information in the README file to explain how far you were able to get. The GTA will compile and run your program for correctness, and look at your solution for design and adherance to the requirements.

Program Compilation

Your program must compile using SmallEiffel 0.75.