Problem 2: NearMiss – A CLASS-BASED PROBLEM

Overview
We all make mistakes when typing a password. Some misses seem more difficult to understand than others. Suppose that our password was “CAT”. Login attempts of “VAT”, “cat”, and even “FW^” are plausible mistakes as each character is either correct or ‘within one key’ of correct. A login attempt of “SoFun2018” is not really a plausible password.

As a separate issue, many login managers disable an account after a certain number of failed attempts to submit its password. The goal here is to prevent hackers from being able to attempt to get in through “brute force” attacks, etc.

Problem
In this problem, you will combine the two ideas above. There are three possible outcomes to a login attempt: success, plausible failure, and non-plausible (major) failure. You will count failures, but you will only count non-plausible failures. The definition of what is plausible will be given through access to a database that could potentially be updated in the future with a more (or less) restrictive definition of what constituted plausibility. Currently, a password is plausible if and only if each character in it is plausible as defined by the database.

In particular, you will be given a series of login attempts for various accounts. In each case, you are to keep track of the attempts and if an account gets too many majorly failing login attempts without an intervening successful one, then you will disable the account. Note: there is no way to re-enable an account in this system.

Input
The first line of input contains a single positive integer L, representing the number of failed – and implausible – login attempts that may be executed before an account is disabled. [Note: a value of 1 for L means that the user gets one mistake, but the second one disables the account unless there was a successful login between them; a value of 0 for L is completely unforgiving – the account is closed immediately upon receipt of an implausible password.] The remaining lines will consist of a series of words, each pair separated by a single space. On each line, the first word is the account name, the second is the current correct password for that account, and the remaining words represent individual attempts to log in to that account. While the password for a given account may change from input-line to input-line, the count of unsuccessful logins is maintained throughout the run of the program, only resetting to zero upon a successful login. The input will be terminated by an account named END – which is not to be processed in any way.

-over-
Output
The output is a series of lines of text reporting on successful logins and account disablements formatted as in the example.

About the class
You must use the PasswordPlausibility class in your solution to this problem. Otherwise, you will not be able to know how to judge which characters are permitted in user names/passwords and/or the plausibility of a login attempt. All of the necessary methods are provided for you (although you may not need to use everything that is provided). Be sure to pay close attention to documentation of the class, including the pre- and post-conditions for all of the methods.

Example (Since account names are not case sensitive, their case in the output does not matter either)

Input  (Note font size shrunk only to fit lines on the page)

2
Dortiz BigPapi BIGPAPI papi34 BigPapi Jeter Boston RedSox BigPapi
jeter2 yankee4Ever tanker$Ever yankee4Ever bogusTry again
PetLover83 CAT cat VAT FW^ SoFun2018 CAT dog
PetLover83 CAT CAT dog CAT
jeter2 HallOfFame yankee4Ever HallOfFame bogusTry
Dortiz BigPapi BigPapi
END

Output

Successful login for Dortiz.
Dortiz account disabled.
Successful login for jeter2.
Successful login for PetLover83.
Successful login for PetLover83.
Successful login for PetLover83.
jeter2 account disabled.

Two Final Notes
This is a pretty restrictive version of plausibility. The idea presented in this problem requires that the password lengths match exactly AND it does not account for transpositions. Thus, both “ACT” and “CATT” are implausible in our initial example. This is a bit unreasonable, but this is a programming contest problem, not a real specification. BESIDES, the whole idea is terribly insecure and should probably NEVER BE USED!