Assignment #2

Problem Solving and Programming in C++
Department of Computer Science
Old Dominion University

Objectives: This assignment will give you an opportunity to explore the process of dividing a **C++** program into modules and using the project support components of a C++ IDE to manage your modules.

General Instructions: Read the problem description below and implement this program in C++. The files for this assignment are provided under the folder "***supporting-files***" for this assignment.

- All of the functions making up this program are complete and in working order except for those marked with //!!comments, which you must implement.
- The major challenge in this assignment is to divide the program into separately compiled modules.
  - A Message module, consisting of files `message.h` and `message.cpp`, containing code dealing specifically with manipulation of entire messages.
  - A `SensitiveWords` module, consisting of files `sensitiveWords.h` and `sensitiveWords.cpp`, containing code dealing specifically with maintaining a list of sensitive words and checking words from a message against that list.
  - There are some functions that are not specifically related to either of these modules, but that contain code needed by them. You should apportion these functions to the modules in a way consistent with the goals of ***high cohesion*** and ***low coupling***.
  - Consult the comments in the provided code for additional details.

Problem description: The Justice Department runs a Witness Protection Plan in which witnesses to crimes are given new identities to protect them from retaliation by the people against whom they testify in court. Experience has shown that many of the protected witnesses are their own worst enemies - often giving away their new locations and identities in misguided attempts to contact relatives and friends and to assure them that all is well.

The Department is experimenting with a new idea of allowing such communications, in email form only, with the idea that employees will inspect the communications first and cut out any potentially dangerous sentences, then send the email from the Justice Department computers so that they cannot be traced back to the witnesses.

Unfortunately, the budget for the pilot project was cut almost as soon as the project commenced. There is insufficient money to actually hire people to read all the email, so an automated solution is sought instead.

Write a **C++** program that, given a list of sensitive words and a message (in plain text form), scans the message for any sentence containing a sensitive word (ignoring differences in upper/lower case). If a sensitive word is found, every character in that sentence (except for line terminators) should be replaced by '@' characters.

For the purposes of this program, a word is a string of consecutive alphanumeric characters bounded in the message by the start or end of the message, and/or by any non-alphanumeric character. A sentence is a string of consecutive words bounded by the start or end of the message, by a paragraph boundary (a line containing zero characters), and/or by one of the punctuation characters: '**.**', '**?**', or '**!**'.

Input

Input to the program is taken from the standard input (`cin`). You will need to change this to make your program read the input data from the text file “input.txt”. Please notice that the input consists of a word list and a message.

A word list consists of zero or more words, one per line and left-justified. Words may be up to 40 characters in length. The end of the word list is signalled by a line containing only the characters **===** (three equal signs).

The word list is immediately followed by a message. A message consists of zero or more lines of text containing up to 80 characters per line. The end of the message is signaled by the end of the input. (Note: when typing input, you can signal end of input with a Ctrl-Z in Windows or a Ctrl-D in Unix/Linux.

Output

Print the message, exactly as it appears in the input except for replacement of sentences by '@' characters as described above. Save your output into a file named “output.txt”

Example

Given the following input which is provided in the input.txt file:

```
John
jane
Smith
Jones
Kansas
court
crime
phone
555
===
Dear Mom,

I just wanted to let you know that I am alive and well. Jane is
well also. I'm glad they were able to relocate us. My
only complaint is that I wish they could have found someplace
more exciting than Kansas for us to live in! If you really need
to contact us, you can do so by telephone. The number is
(757) 555-0478, but don't tell anyone.

Love,
the new John Smith
```

The output should look as follow:

```
Dear Mom,

I just wanted to let you know that I am alive and well.@@@@@@@@
@@@@@@@@@@ I'm glad they were able to relocate us.@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ If you really need
to contact us, you can do so by telephone.@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@@@@@
@@@@@@@@@@@@@@@@@@
```

Notes

It is important to note that the output must match the given format **exactly**. In all assignments, whenever I speak of a line of output, that line must be properly terminated (by `"\n"` or `endl`).

Submission notes:

- Submit all files from your project, especially the .cpp, .cbp and .h
- Zip all the .cpp and .h files and name it as "Assg2_cslogin", where the cslogin is your login ID for the computers at the Department of Computer Science at ODU.
- Submit the zipped file in the respective Blackboard link.