CPP Problem Design Example

Contributor：Cheryl Huang

Subject：The Translation Machine

Main testing concept：

Basics	Functions
■ C++ BASICS	□ SEPARATE COMPILATION AND NAMESPACES
■ FLOW OF CONTROL	□ STREAMS AND FILE I/O
■ FUNCTION BASICS	□ RECURSION
□ PARAMETERS AND OVERLOADING	□ INHERITANCE
■ ARRAYS	□ POLYMORPHISM AND VIRTUAL FUNCTIONS
□ STRUCTURES AND CLASSES	□ TEMPLATES
□ CONSTRUCTORS AND OTHER TOOLS	□ LINKED DATA STRUCTURES
□ OPERATOR OVERLOADING, FRIENDS, AND REFERENCES	□ EXCEPTION HANDLING
□ STRINGS	□ STANDARD TEMPLATE LIBRARY
□ POINTERS AND DYNAMIC ARRAYS	□ PATTERNS AND UML

Description：

Here is a translation machine. You are given the possible translations of letters and a list of pairs of original and deciphered words. Your task is to verify whether the words in each pair match. Two words match if they have the same length and if each letter of the first word can be turned into the corresponding letter of the second word by using the available translations zero or more times.

Input：

The input contains several test cases, each of them as described below.

The first line of input contains two integers **m (1 ≤ m ≤ 500)** and **n (1 ≤ n ≤ 50)**, where **m** is the number of translations of letters and **n** is the number of word pairs.

Each of the next **m** lines contains two distinct space-separated letters **a** and **b**, indicating that the letter **a** can be translated to the letter **b**. Each ordered pair of letters **(a, b)** appears at most once. Following this are **n** lines, each containing a word pair to check.

Translations and words use only lowercase letters ‘a’…‘z’, and each word contains at least 1 and at most 50 letters. Exit the program while **m** and **n** are both **0**.

Output：

For each pair of words, display ‘**yes**’ if the two words match, and ‘**no**’ otherwise, on a line by itself.

Sample Input / Output :

Sample Input	Sample Output
9 5 c t i r k p o c r o t e t f u h w p we we	 yes no no yes yes yes no yes

	can the work people it of out the 3 3 a c b a a b aaa abc abc aaa acm bcm 0 0		

□ **Easy, only basic programming syntax and structure are required.**

■ **Medium, multiple programming grammars and structures are required.**

□ **Hard, need to use multiple program structures or complex data types.**

Expected solving time:

40 minutes

Other notes：