## Strength of Relation

Strength of relation between two words S1 and S2 is defined by the set of position of common characters of S1 and S2, it is defined as Sum(pos(c) in S1 * pos(c) in S2) where c is the character which is present in both S1 and S2. Given a collection of words S and a test string W, write a C++ program to find all strings in S with highest strength of relation with W. When there are many strings with highest strength of relation, print all strings in the order of input.

For example, consider the words thing and night, then the strength of relation (S1 to S2) is 1*5+2*4+3*2+4*1+5*3 = 38.

Given a collection of five words S as apple, tiny, hit, thing and get and W as night, you should print thing which has the maximum strength of relation.

**Hint:**

This problem can be quickly solved with string in STL. STL is Standard Template Library that has got generic functions, classes and algorithms.

- To use string, add #include<string>
- We can create object and array of objects for string
- length is a member function of string that gives the number of characters in it
- You can access ith character of a string object s as s[i] – Subscript operator is overloaded
- find is a member function that takes a character or string as argument and returns the position of occurrence if present and -1 otherwise

**Sample Code using String in STL**

```
#include<iostream>

using namespace std;

#include<string>

int main()

{

string s1,s2;

int l,pos;

cin>>s1;

cin>>s2;

//get length of s1

l=s1.length();

//check if first character of s1 is present in s2

pos=s2.find(s1[0]);

cout<<pos;

}
```

**Input Format**

## Problem Statistics

**Author:** M Janaki Meena

**Solved By:** 0

View leaderboard

First line contains the number of words ‘n’ in the collection, S

Next ‘n’ lines contain the words in S

Next line contains the word W

**Output Format**

Print all the words in S that has got maximum strength of relation

Theme: Terminal Language C++ Font size: 16

```
        {
            strength += j*pos;
        }
    }
    if (strength > max_strength)
    {
        max_strength = strength;
        word.erase(word.begin(),word.end());
        word.push_back(array_s[i]);
    }
    else if (strength == max_strength)
    {
        word.push_back(array_s[i]);
    }
}

for(i=0;i<word.size();i++)
{
    cout<<word[i]<<endl;

```

Use custom I/O

Run Code Save Code Pause Test

## Status:

Success your code has passed all test cases!!