Shortest Word Distance

Given a list of words and two words word1 and word2, return the shortest distance between these two words in the list.

For example,
Assume that words = ["practice", "makes", "perfect", "coding", "makes"].

Given word1 = “coding”, word2 = “practice”, return 3.
Given word1 = "makes", word2 = "coding", return 1.

Note:
You may assume that word1 does not equal to word2, and word1 and word2 are both in the list.

Shortest Word Distance II

This is a follow up of Shortest Word Distance. The only difference is now you are given the list of words and your method will be called repeatedly many times with different parameters. How would you optimize it?

Design a class which receives a list of words in the constructor, and implements a method that takes two words word1 and word2 and return the shortest distance between these two words in the list.

For example,
Assume that words = ["practice", "makes", "perfect", "coding", "makes"].

Given word1 = “coding”, word2 = “practice”, return 3.
Given word1 = "makes", word2 = "coding", return 1.

Note:
You may assume that word1 does not equal to word2, and word1 and word2 are both in the list.

Shortest Word Distance III

This is a follow up of Shortest Word Distance. The only difference is now word1 could be the same as word2.

Given a list of words and two words word1 and word2, return the shortest distance between these two words in the list.

word1 and word2 may be the same and they represent two individual words in the list.

For example,
Assume that words = ["practice", "makes", "perfect", "coding", "makes"].

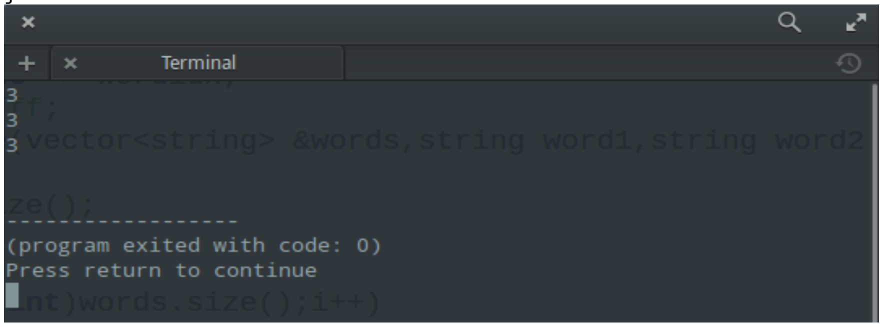
Given word1 = “makes”, word2 = “coding”, return 1.

Given word1 = "makes", word2 = "makes", return 3.

Note:
You may assume word1 and word2 are both in the list.

```
// author:zzw
#include <iostream>
#include <vector>
#include <string>
#include <map>
#include <algorithm>
using namespace std;
map<string,vector<int> > wordidx;
#define INT_MAX 0xffff;
int shortestDistance(vector<string> &words,string word1,string word2)
{
	int res=words.size();
	int idx1=-1;
	int idx2=-1;
	for (int i=0;i<(int)words.size();i++)
	{
		if(words[i]==word1)
		{
			idx1=i;
			if(idx2!=-1) res = min(res,idx1-idx2);
		}
		else if(words[i]==word2)
		{
			idx2=i;
			if(idx1!=-1) res = min(res,idx2-idx1);
		}
	}
	return res;
}

void WordDistance(vector<string> &words)
{
	for(int i=0;i<(int)words.size();i++)
	{
		wordidx[words[i]].push_back(i);
	}
}

int shortestDistance2(string word1,string word2)
{
	int res = INT_MAX;
	vector<int> idx1 = wordidx[word1];
	vector<int> idx2 = wordidx[word2];
	int m=idx1.size();int n=idx2.size();
	int i=0;int j=0;
	while(i<m && j<n)
	{
		res = min(res,abs(idx1[i]-idx2[j]));
		if(idx1[i]>idx2[j]) j++;
		else i++;
	}
	return res;
}

int shortest(vector<string> &words, string word1)
{
	int pre=-1;
	int res = words.size();
	for(int i=0;i<(int)words.size();i++)
	{
		if(words[i]==word1)
		{
			if(pre!=-1) res = min(res,i-pre);
```

```
                        pre=i;
                }
        }
        return res;
}

int shortestDistance3(vector<string> &words,string word1,string word2)
{
        int res=words.size();
        int idx1=-1;
        int idx2=-1;
        if(word1==word2)
        {
                return shortest(words,word1);
        }

        for (int i=0;i<(int)words.size();i++)
        {
                if(words[i]==word1)
                {
                        idx1=i;
                        if(idx2!=-1) res = min(res,idx1-idx2);
                }
                else if(words[i]==word2)
                {
                        idx2=i;
                        if(idx1!=-1) res = min(res,idx2-idx1);
                }
        }
        return res;
}


int main(int argc,char *argv[])
{
        // test shortestDistance1
        string str[] = {"practice", "makes", "perfect", "coding", "makes"};
        vector<string> words;
        for (int k=0;k<5;k++)
                words.push_back(str[k]);
        int ans = shortestDistance(words,"coding","practice");
        cout << ans << endl;
        // test shortestDistance2
        WordDistance(words);
        ans = shortestDistance2("coding","practice");
        cout << ans << endl;
        // test shortestDistance3
        ans = shortestDistance3(words,"makes","makes");
        cout << ans << endl;
        return 0;
}
```



```
3
3
3


------------------
(program exited with code: 0)
Press return to continue
```