290. Word Pattern

Given a pattern and a string str, find if str follows the same pattern.

Here **follow** means a full match, such that there is a bijection between a letter in pattern and a **non-empty** word in str.

Examples:

1. pattern = "abba", str = "dog cat cat dog" should return true.
2. pattern = "abba", str = "dog cat cat fish" should return false.
3. pattern = "aaaa", str = "dog cat cat dog" should return false.
4. pattern = "abba", str = "dog dog dog dog" should return false.

Notes:

You may assume pattern contains only lowercase letters, and str contains lowercase letters separated by a single space.

291. Word Pattern II

Given a pattern and a string str, find if str follows the same pattern.

Here **follow** means a full match, such that there is a bijection between a letter in pattern and a **non-empty** substring in str.

Examples:

1. pattern = "abab", str = "redblueredblue" should return true.
2. pattern = "aaaa", str = "asdasdasdasd" should return true.
3. pattern = "aabb", str = "xyzabcxzyabc" should return false.

Notes:

You may assume both pattern and str contains only lowercase letters.

```
#include<sstream>
```

```
#include<iostream>
#include<algorithm>
#include<string>
#include<vector>
#include<map>
using namespace std;
//author :DemonMikalis
bool WordPattern(string pattern, string str)
{
	vector<string> words;
	string word;
	stringstream ss(str);
	while(ss>>word)
		words.push_back(word);
	map<char,int> mappa;
	map<char,int>::iterator it1;
	map<string,int> mapstr;
	map<string,int>::iterator it2;
	for(int i=0;i<pattern.size();++i)
	{
		it1 = mappa.find(pattern[i]);
		it2 = mapstr.find(words[i]);
		if(it1!=mappa.end())
		{
			if(words[it1->second]!=words[i])
			return false;
		}else if (it2 !=mapstr.end())
		{
			if(pattern[it2->second]!=pattern[i])
			return false;
		}else{
			mappa.insert(make_pair<char,int>(pattern[i],i));
			mapstr.insert(make_pair<string,int>(words[i],i));
		}
	}
	return true;
}

bool WordPatternII(string pattern, int k, string str, int r, map<char,string> &m)
{
```

```
        if(pattern.size()==k && str.size()==r) return true;
        if(pattern.size()==k || str.size()==r) return false;
        char c=pattern[k];
        for(int i=r;i<str.size();i++)
        {
                string t = str.substr(r,i-r+1);
                if(m.count(c) && m[c]==t)
                {
                        if (WordPatternII(pattern,k+1,str,i+1,m)) return true;
                }else if (!m.count(c)){
                        bool b = false;
                        map<char,string>::iterator it;
                        for(it=m.begin();it!=m.end();it++)
                        {
                                string tmp = it->second;
                                if(t==tmp) b=true;
                        }
                        if(b==false)
                        {
                                m[c]=t;
                                if (WordPatternII(pattern,k+1,str,i+1,m)) return true;
                                m.erase(c);
                        }
                }
        }
        return false;
}

int main(int argc, char *argv[])
{
        // test 1
        bool ans=WordPattern("abba","dog cat cat dog");
        bool ans2=WordPattern("abba","dog cat cat fish");
        bool ans3=WordPattern("aaaa","dog cat cat dog");
        bool ans4=WordPattern("aaab","dog dog dog cat");
        cout<<ans<<ans2<<ans3<<ans4<<endl;
        // test 2
        map<char,string> m;
        bool b1 = WordPatternII("abab",0,"redblueredblue",0,m);
        m.clear();
```

```
        bool b2 = WordPatternII("aaaa",0,"asdasdasdasd",0,m);
        m.clear();
        bool b3 = WordPatternII("aabb",0,"xyzabcxzyabc",0,m);
        m.clear();
        bool b4 = WordPatternII("abab",0,"xyzabcxyzabc",0,m);
        cout<<b1<<b2<<b3<<b4<<endl;
        return 0;
}
```

OP: