#include <bits/stdc++.h>

using namespace std;

class FD {

public:

string alphaStr, betaStr;

public:

FD(string fd) {

alphaStr = "", betaStr = "";

bool currentlyAlpha = true;

for(int i = 0; i < fd.length(); ++i) {

if(fd[i] == '-' || fd[i] == '>') {

currentlyAlpha = false;

} else if (currentlyAlpha) {

alphaStr += fd[i];

} else {

betaStr += fd[i];

}

}

}

};

string UnionString(string a, string b) {

bool visited[26] = {false};

for(int i = 0; i < a.length(); ++i) {

visited[a[i] - 'A'] = true;

}

for(int i = 0; i < b.length(); ++i) {

visited[b[i] - 'A'] = true;

}

string ans = "";

for(int i = 0; i < 26; ++i) {

if(visited[i]) ans += 'A' + i;

}

return ans;

}

string GetSubstr(string str, int a, int b) {

string ans = "";

for(int i = a; i <= b && a <= b; ++i) {

ans += str[i];

}

return ans;

}

bool CheckForSubstring(string a, string b) {

if(a.length() < b.length()) return false;

bool allFlag = false;

for(int i = 0; i < a.length() - b.length() + 1; ++i) {

bool flag = true;

for(int j = 0; j < b.length(); ++j) {

flag = flag && (a[i + j] == b[j]);

}

allFlag = allFlag || flag;

}

return allFlag;

}

string getClosure(map<string, string> fd, string input) {

string ans = input;

for(int k = 0; k < fd.size(); ++k) {

for(int i = 0; i < ans.length(); ++i) {

for(int j = 0; j < ans.length(); ++j) {

string a = GetSubstr(ans, i, j);

if(fd.find(a) != fd.end()) {

// cout << fd[a] << "->" << a << endl;

ans = UnionString(ans, fd[a]);

}

}

}

}

return ans;

}

int main() {

int countAttrSet, countFd;

cin >> countAttrSet >> countFd;

char tempChar;

string tempStr, attrSetStr = "";

vector<char> attrSet;

FD \*temp;

vector<string> alphas, betas;

map<string, string> fd;

for(int i = 0; i < countAttrSet; ++i) {

cin >> tempChar;

attrSetStr += tempChar;

attrSet.push\_back(tempChar);

}

for(int i = 0; i < countFd; ++i) {

cin >> tempStr;

temp = new FD(tempStr);

alphas.push\_back(temp->alphaStr);

betas.push\_back(temp->betaStr);

fd[temp->alphaStr] = temp->betaStr;

}

int total = 0;

vector<string> candidateKeys;

for(int len = 0; len < attrSetStr.length(); ++len) {

for(int j = 0; j < attrSetStr.length() - len; ++j) {

string subs = GetSubstr(attrSetStr, j, j + len);

string closure = getClosure(fd, subs);

if (closure.length() == attrSetStr.length()) {

bool flag = true;

vector<string>::iterator x;

for(x = candidateKeys.begin(); x != candidateKeys.end(); ++x) {

flag = flag && (!(CheckForSubstring(subs, \*x) || CheckForSubstring(\*x, subs)));

}

if(flag) candidateKeys.push\_back(subs);

}

cout << subs << "->" << closure << endl;

total += (int) pow(2, closure.length());

}

}

cout << "Total Functional Dependencies: " << total << endl;

cout << "Candidate Keys: ";

vector<string>::iterator x;

for(x = candidateKeys.begin(); x != candidateKeys.end(); ++x) {

cout << \*x << ", ";

}

cout << endl;

}