```
//
//  main.cpp
//  C++ Terminal
//
//  Created by Jiaqi Li on 2019/12/29.
//  Copyright © 2019 Jiaqi Li. All rights reserved.
//
//  PARSING PROGRAM TO PARSE HRML'S TAGS AND QUERIES WITH STRUCTS AND
 STRING COMMANDS
//  HRML IS A MOCK-HTML LANGUAGE
//  The programm supports up to 3 attributes (dynamic programming not
 learned in C++ as of 2019/12/29)
//
//  ARCHITECTURE (without for loops / i/o)
//      parsing tags: use getline and stringstream
//      saving tags: get tag attribute and content into appropriate line of
 struct
//      Creating tag heiarchy uses bool and a mutated bubbling algorithm
//      parsing queries: use string.find to locate ~,find the number before
 ~
//          store and compare the attribute with respective line in struct
//      output: printout content if found, if not print 'Not Found!'
//  *if content of an attribute is a number it must be greater than zero

#include <iostream>
#include <string>
#include <sstream>
#include <cmath>
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;

//////////////////////////////////////////////////////////////////////////////
 ////
// struct for saving tags
struct tagbook{
    string tag, attribute1, content1, attribute2, content2, attribute3,
     content3;
    int start, end;
    bool status; // 1 = closed 0 = open
};


//////////////////////////////////////////////////////////////////////////////
 /////
int main() {


     /////////////////////////////////////////////////////////////////////////
     ////////////
    // input block
    int N, Q;
    cin >> N >> Q;
    ////////////////////////////////////////////////////////////////////
    ////////////
   // tag parsing block
   //Initialize Vars
   struct tagbook arr[N/2];
   string tag;
   int tagsize;
   int counter=0;

   //For loop to parse and record the tags
   for (int i = 0; i < N; i++){

       //format of ss: <tagn; attribute; = ; "content">, max three
        attributes
       string tagname, tagattribute1, k, tagcontent1, tagattribute2,
        tagcontent2, tagattribute3, tagcontent3;
       if (i==0){
           cin.ignore();
       }
       getline(cin, tag);
       stringstream ss(tag);
       ss >> tagname >> tagattribute1 >> k >> tagcontent1 >> tagattribute2
        >> k >> tagcontent2 >> tagattribute3 >> k >> tagcontent3;

       //in the first run check tag length
       if (i==0){
           tagsize = tagname.size()-1;
       }

       //if it's an opening tag, erase the '<>'&quotation marks, load the
        attributes and contents
       if (tagname[1] != '/'){

           if (tagcontent1.size()!=0){
           tagname.erase(tagname.begin());
           }
           else{
               tagname = tagname.substr(1, tagname.size() - 2);
           }

           if (tagcontent1.size()!=0){
               if (tagcontent1.find(">")!= -1){
                   tagcontent1.pop_back();
               }
               tagcontent1 = tagcontent1.substr(1, tagcontent1.size() - 2);
           }
           if (tagcontent2.size()!=0){
               if (tagcontent2.find(">")!= -1){
                   tagcontent2.pop_back();
                   }
               tagcontent2 = tagcontent2.substr(1, tagcontent2.size() - 2);
           }
           if (tagcontent3.size()!=0){
               if (tagcontent3.find(">")!= -1){
                tagcontent3.pop_back();
            }
            tagcontent3 = tagcontent3.substr(1, tagcontent3.size() - 2);
        }
        arr[counter].attribute1 = tagattribute1;
        arr[counter].content1 = tagcontent1;
        arr[counter].attribute2 = tagattribute2;
        arr[counter].content2 = tagcontent2;
        arr[counter].attribute3 = tagattribute3;
        arr[counter].content3 = tagcontent3;
        arr[counter].tag = tagname;
        arr[counter].start = i; //save order of tag entry
        arr[counter].status = 0;
        counter = counter + 1;
    }

    //if it's an closing tag, erase the '<>' & load the tag name
    else{
        tagname = tagname.substr(2, tagname.size() - 3);
        for (int l = 0; l < N/2; l++){
            if (arr[l].tag.compare(tagname) == 0){
                arr[l].end = i;
            }
        }
        }
}

//create tag heiarchy with bool and bubbling algorithm
char dot = '.';

//first work on tags that close immediately after it opens
//search backwards for tags that are open and whose closing tags are
 after that searched for
//add those tag names to the front of the current tag name
for (int j = 0; j< N/2 ; j++){
    if (arr[j].end-arr[j].start == 1){
        for (int r = j-1; r > -1; r--){
            if (arr[r].status == 0 && arr[r].end > arr[j].end){
                arr[j].tag = arr[r].tag + dot + arr[j].tag;
            }
        }
        arr[j].status = 1;
    }
}

//then work on the rest
//same algorithm
for (int j = (N/2)-1; j > -1 ; j--){
    if (arr[j].status == 0){
        for (int r = j-1; r > -1; r--){
            if (arr[r].status == 0 && arr[r].end > arr[j].end){
                arr[j].tag = arr[r].tag + dot + arr[j].tag;
            }
        }
        arr[j].status = 1;
        }
    }


     ////////////////////////////////////////////////////////////////////////
     ////////
    // query parsing block
    string query;
    string queryattribute;

    for (int i = 0; i < Q; i++){
    getline(cin, query);
        queryattribute = query;
        //trimming block, query retains tag name, querycontent retains
         attribute
        if (query.find("~")<0){
            cout << "Not Found!" << endl;
        }
        else{
            size_t location = query.find("~")+1;
            int eraserange = location;
            queryattribute.erase(queryattribute.begin()+0,
             queryattribute.begin()+eraserange);
            query.erase(query.begin()+location-1, query.end());

            //comparison and output block
            int querycounter=0;
            for (int p = 0; p < N/2; p++){
                if (arr[p].tag.compare(query) == 0){
                    if (arr[p].attribute1.compare(queryattribute) == 0){
                        cout  << arr[p].content1 << endl;
                    }
                    else if (arr[p].attribute2.compare(queryattribute) ==
                     0){
                        cout  << arr[p].content2 << endl;
                    }
                    else if (arr[p].attribute3.compare(queryattribute) ==
                     0){
                        cout  << arr[p].content3 << endl;
                    }
                    else{
                        cout << "Not Found!" << endl;
                    }
                }
                else {
                    querycounter = querycounter+1;
                }
            }
            if (querycounter == N/2){
                cout << "Not Found!" << endl;
            }
        }
    }
    return 0;
}
```