NAJPF - Pattern Find

[#kmp-algorithm](https://www.spoj.com/problems/tag/kmp-algorithm)

Your task is so simple given a string and a pattern. You find the pattern on the given string. If found print how many time found the pattern and their index. Otherwise print ‘Not Found’

**Input:**

The input   line consists of a number T (1 ≤ T ≤ 50) test cases.

For each case given two string number  A,B. the string and the pattern  1 ≤|A|, |B| ≤10^6

All character will be lower case Latin character.  And |  | is the length of string.

**Output:**

For each case print the number  (found pattern from the given string) next line there position And Otherwise print 'Not Found' without quota.  
There will a blank line between two cases.

**Sample:**

|  |  |
| --- | --- |
| Input | Output |
| 3 ababab ab aaaaa bbb aafafaasf aaf | 3 1 3 5  Not Found  1 1 |

Hints:

Here all index is 1 base.

Solution:

1. #include<bits/stdc++.h>
2. #define pb push\_back
3. #define int long long int
4. #define INF 1e18
5. #define vec vector<int>
6. #define REP(i,a,b) for(i=a;i<b;i++)
7. using namespace std;
8. int mod=1e9+7;
9. main()
10. {
11. ios\_base::sync\_with\_stdio(false);
12. [cin](http://www.opengroup.org/onlinepubs/009695399/functions/cin.html).tie(NULL);
13. [cout](http://www.opengroup.org/onlinepubs/009695399/functions/cout.html).tie(NULL);
14. int t;
15. cin>>t;
16. while(t--){
17. string M,X;
18. cin>>M>>X;
19. int x=X.length();
20. int p=31,i;
21. int m=M.length();
22. vector<int> power(1000001,0),has(m+1,0),occ;
23. power[0]=1;
24. for(i=1;i<1000001;i++)
25. power[i]=(power[i-1]\*p)%mod;
26. for(i=0;i<m;i++)
27. has[i+1]=(has[i]+(M[i]-'a'+1)\*power[i]%mod)%mod;
28. int hs=0,kh=0;
29. for(i=0;i<x;i++)
30. hs=(hs+(X[i]-'a'+1)\*power[i]%mod)%mod;
31. for(i=0;i<m+1-x;i++)
32. {
33. int curr=(has[i+x]-has[i]+mod)%mod;
34. if(curr==hs\*power[i]%mod)
35. occ.pb(i+1);
36. }
37. if(occ.size()==0)
38. cout<<"Not Found**\n**";
39. else
40. {
41. cout<<occ.size()<<"**\n**";
42. for(int ele:occ)
43. cout<<ele<<" ";
44. cout<<"**\n**";
45. }
46. }
47. }