// Trie 数

const int maxnode = 4000010;

const int sigma\_size = 26;

struct Trie {

    int ch[maxnode][sigma\_size];

    int sz, val[maxnode];

void clear()

{sz=0;memset(ch[0],0,sizeof(ch[0]));}

    // 一开始只有一个根节点0

    // 点的编号: 0 ~ sz

    int idx(char c){return c-'a';}

    void insert(char \*s, int v) {

        int u = 0, n = strlen(s);

        for(int i=0;i<n;i++) {

            int id = idx(s[i]);

            if(!ch[u][id]) {

                ++sz;

                memset(ch[sz],0,sizeof(ch[sz]));

                val[sz]=0;

                ch[u][id]=sz;

            }

            u = ch[u][id];

        }

        val[u] += v;

    }

    int search(char \*s) {

        int u = 0, n = strlen(s);

        for(int i=0;i<n;i++) {

            int id = idx(s[i]);

            if(ch[u][id]==0)

                return 0;

            u = ch[u][id];

        }

        return val[u];

    }

};

Trie trie;

// KMP 字符串匹配

char P[maxn]; // Pattern 短串

char T[maxn]; // Text 长串

int f[maxn];

void getFail(char\* P,int\* f) {

//字符串p自我匹配

int m = strlen(P);

f[0] = f[1] = 0;

for(int i = 1; i < m; i++) {

int j = f[i];

while(j && P[i]!=P[j])

j = f[j];

f[i+1] = P[i]==P[j] ? j+1 : 0;

}

}

void Find(char\* T, char\* P, int\* f) {

//p去匹配字符串T

int n = strlen(T), m = strlen(P);

getFail(P, f); //得出部分匹配表

int j = 0;

//j:短串的下标 i: 长串下标

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

while (j && P[j] != T[i])

j = f[j];

if (P[j] == T[i]) j++;

if(j == m)

printf("%d ", i-m+1);

}

puts("");

}

int main() {

// c++ getline(cin, P)

// c gets(P)

while (gets(P))

{gets(T); Find(T, P, f);}

}