// KMP模板

int next[1000005] = {0};

/// "部分匹配值"就是"前缀"和"后缀"的最长的共有元素的长度

/// 移动位数 = 已匹配的字符数 - 对应的部分匹配值

void makeNext (const char p[])

{

int q, k;// q:模版字符串下标；k:最大前后缀长度

int m = strlen(p); // 模板字符串长度

next[0] = 0; // 模板字符串的 第一个字符的最大的前后缀长度为0

for(q = 1, k = 0; q < m ; ++q )

{// 第二个字符开始 依次计算每一个字符对应的next值

while ( k > 0 && p[q] != p[k] )

// 递归求出 p[0] .. p[q] 的最大的相同前后缀 长度k

k = next[k-1]; // 往前回溯

if( p[q] == p[k] ) //如果相等，那么最大相同前后缀长度加1

k++;

next[q] = k;

// cout << q << ' ' << k << endl;

}

}

int kmp(char t[],char p[])

{

int n = strlen(t), m = strlen(p);

int cnt = 0;

makeNext(p);

for(int i = 0, q = 0; i < n; ++i)

{

while( q > 0 && p[q] != t[i])

q = next[q-1];

if( p[q] == t[i] )

q++;

if( q == m)

cnt++;

//cout << i << ' ' << q << endl;

}

return cnt;

}

char t[ 1000005 ], p[1000005 ];

scanf("%s", p);

scanf("%s", t);

printf("%d\n",kmp(t, p));

/\* MAKENEXT的优化： GETNEXT:

void getNext(const char \*pStr)

{

int j = 0, j\_next = -1, pLen = strlen(pStr);

next1[j] = j\_next;

int mLen = pLen - 1;

while (j < mLen)

{

if (pStr[++j] != pStr[++j\_next])

{

next1[j] = j\_next;

while (j\_next != -1 && pStr[j] != pStr[j\_next]) j\_next = next1[j\_next];

}

else next1[j] = next1[j\_next];

}

}

int kmp(char \*source, char\* pattern) {

int i = 0;

int j = 0;

int p\_len = strlen( pattern );

int s\_len = strlen( source );

//makeNext(pattern);

getNext(pattern);

while (j < p\_len && i < s\_len) {

if (j == -1 || source[i] == pattern[j]) {

i++;

j++;

}

else {

j = next1[j];

}

}

if (j < strlen( pattern) )

return -1;

else

return i - strlen( pattern );

}