char strt[1000009];

char strp[1000009];

int lps[1000009];

void lpscalc()

{

int j = 0; // length of the previous longest prefix suffix

int i;

lps[0] = 0; // lps[0] is always 0

i = 1;

int plen = strlen(strp);

// the loop calculates lps[i] for i = 1 to M-1

while (i < plen)

{

if (strp[j] == strp[i])

{

lps[i] = j+1;

i +=1;

j+=1;

}

else // (pat[i] != pat[j])

{

if (j != 0)

{

j = lps[j-1];

}

else // if (j == 0)

{

lps[i] = 0;

i++;

}

}

}

}

int nummatch()

{

int cnt = 0;

int pat\_index = 0, text\_index = 0;

int plen = strlen(strp);

int tlen = strlen(strt);

if(plen == 0)

{

return 0;

}

while(text\_index < tlen)

{

// if characters match, look for next character match

if(strp[(pat\_index)] == strt[(text\_index)])

{

pat\_index++;

text\_index++;

// indicates that complete pattern has matched

if(pat\_index == plen)

{

cnt++;

pat\_index = lps[pat\_index-1];

}

}

// if the characters do not match, don't go back in the text. Just adjust the pattern\_index

else

{

if(pat\_index != 0)

{

pat\_index = lps[pat\_index-1];

}

else

{

text\_index++;

}

}

}

return cnt;

}