#include <iostream>

#include <vector>

Using namespace std;

// Function to compute the Longest Prefix Suffix (LPS) array

Void computeLPS(const string &pattern, vector<int> &lps) {

Int length = 0; // length of the previous longest prefix suffix

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

Int I = 1;

While (I < pattern.length()) {

If (pattern[i] == pattern[length]) {

Length++;

Lps[i] = length;

I++;

} else {

If (length != 0) {

// Backtrack the length

Length = lps[length – 1];

} else {

Lps[i] = 0;

I++;

}

}

}

}

// Function to implement the KMP algorithm

Void KMPAlgorithm(const string &text, const string &pattern) {

Int n = text.length();

Int m = pattern.length();

// Create the LPS array

Vector<int> lps(m);

computeLPSArray(pattern, lps);

int I = 0; // index for text

int j = 0; // index for pattern

while (I < n) {

if (pattern[j] == text[i]) {

i++;

j++;

}

If (j == m) {

Cout << “Pattern found at index “ << I – j << endl;

J = lps[j – 1]; // Get the next position from LPS array

} else if (I < n && pattern[j] != text[i]) {

If (j != 0) {

J = lps[j – 1]; // Backtrack in the pattern

} else {

I++;

}

}

}

}

Int main() {

String text, pattern;

Cout << “Enter the text: “;

Cin >> text;

Cout << “Enter the pattern to search: “;

Cin >> pattern;

KMPAlgorithm(text, pattern);

Return 0;

}