**Code:**

#include <bits/stdc++.h>

#include <chrono>

using namespace std;

using namespace std::chrono;

// Naive String Matching Algorithm

void naiveStringMatching(const string& text, const string& pattern) {

int n = text.length();

int m = pattern.length();

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

int j;

for (j = 0; j < m; ++j)

if (text[i + j] != pattern[j])

break;

if (j == m)

cout << "Naive: Pattern found at index " << i << endl;

}

}

// Rabin-Karp Algorithm

void rabinKarp(const string& text, const string& pattern) {

int n = text.length();

int m = pattern.length();

int prime = 101; // A prime number

int patternHash = 0;

int textHash = 0;

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

patternHash += pattern[i] \* pow(prime, i);

textHash += text[i] \* pow(prime, i);

}

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

if (patternHash == textHash) {

int j;

for (j = 0; j < m; ++j)

if (text[i + j] != pattern[j])

break;

if (j == m)

cout << "Rabin-Karp: Pattern found at index " << i << endl;

}

if (i < n - m) {

textHash = (textHash - text[i] + text[i + m] \* pow(prime, m - 1)) \* prime;

}

}

}

// Knuth-Morris-Pratt Algorithm

void buildLPS(const string& pattern, int\* lps) {

int m = pattern.length();

int len = 0;

lps[0] = 0;

int i = 1;

while (i < m) {

if (pattern[i] == pattern[len]) {

len++;

lps[i] = len;

i++;

} else {

if (len != 0)

len = lps[len - 1];

else {

lps[i] = 0;

i++;

}

}

}

}

void kmp(const string& text, const string& pattern) {

int n = text.length();

int m = pattern.length();

int\* lps = new int[m];

buildLPS(pattern, lps);

int i = 0, j = 0;

while (i < n) {

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

i++;

j++;

}

if (j == m) {

cout << "KMP: Pattern found at index " << i - j << endl;

j = lps[j - 1];

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

if (j != 0)

j = lps[j - 1];

else

i++;

}

}

delete[] lps;

}

int main() {

string text, pattern;

cout << "Enter the text: ";

getline(cin, text);

cout << "Enter the pattern: ";

getline(cin, pattern);

auto start = high\_resolution\_clock::now();

naiveStringMatching(text, pattern);

auto stop = high\_resolution\_clock::now();

auto naiveDuration = duration\_cast<microseconds>(stop - start);

start = high\_resolution\_clock::now();

rabinKarp(text, pattern);

stop = high\_resolution\_clock::now();

auto rabinKarpDuration = duration\_cast<microseconds>(stop - start);

start = high\_resolution\_clock::now();

kmp(text, pattern);

stop = high\_resolution\_clock::now();

auto kmpDuration = duration\_cast<microseconds>(stop - start);

cout << "Naive String Matching Time: " << naiveDuration.count() << " microseconds" << endl;

cout << "Rabin-Karp Time: " << rabinKarpDuration.count() << " microseconds" << endl;

cout << "KMP Time: " << kmpDuration.count() << " microseconds" << endl;

return 0;

}

**Output:**

