**Rabin Karp :-**

#include <bits/stdc++.h>

#include <cstring>

using namespace std;

#define d 256

void search(char pat[], char txt[], int q)

{

    int M = strlen(pat);

    int N = strlen(txt);

    int i, j;

    int p = 0;

    int t = 0;

    int h = 1;

    for (i = 0; i < M - 1; i++)

        h = (h \* d) % q;

    for (i = 0; i < M; i++)

    {

        p = (d \* p + pat[i]) % q;

        t = (d \* t + txt[i]) % q;

    }

    for (i = 0; i <= N - M; i++)

    {

        if (p == t)

        {

            bool flag = true;

            for (j = 0; j < M; j++)

            {

                if (txt[i + j] != pat[j])

                {

                    flag = false;

                    break;

                }

            }

            if (j == M)

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

        }

        if (i < N - M)

        {

            t = (d \* (t - txt[i] \* h) + txt[i + M]) % q;

            if (t < 0)

                t = (t + q);

        }

    }

}

int main()

{

    string text;

    string pattern;

    cout << "Enter the text : ";

    getline(cin, text);

    cout << "Enter the pattern : ";

    getline(cin, pattern);

    char txt[text.length() + 1];

    char pat[pattern.length() + 1];

    strcpy(txt, text.c\_str());

    strcpy(pat, pattern.c\_str());

    int q = 101;

    search(pat, txt, q);

    return 0;

}

**Output:**

Text

Description automatically generated

**KMP :-**

#include <bits/stdc++.h>

#include <cstring>

using namespace std;

void computeLPSArray(char \*pat, int M, int \*lps);

void KMPSearch(char \*pat, char \*txt)

{

    int M = strlen(pat);

    int N = strlen(txt);

    int lps[M];

    computeLPSArray(pat, M, lps);

    int i = 0;

    int j = 0;

    while (i < N)

    {

        if (pat[j] == txt[i])

        {

            j++;

            i++;

        }

        if (j == M)

        {

            printf("Found pattern at index %d ", i - j);

            j = lps[j - 1];

        }

        else if (i < N && pat[j] != txt[i])

        {

            if (j != 0)

                j = lps[j - 1];

            else

                i = i + 1;

        }

    }

}

void computeLPSArray(char \*pat, int M, int \*lps)

{

    int len = 0;

    lps[0] = 0;

    int i = 1;

    while (i < M)

    {

        if (pat[i] == pat[len])

        {

            len++;

            lps[i] = len;

            i++;

        }

        else

        {

            if (len != 0)

            {

                len = lps[len - 1];

            }

            else

            {

                lps[i] = 0;

                i++;

            }

        }

    }

}

int main()

{

    string text;

    string pattern;

    cout << "Enter the text : ";

    getline(cin, text);

    cout << "Enter the pattern : ";

    getline(cin, pattern);

    char txt[text.length() + 1];

    char pat[pattern.length() + 1];

    strcpy(txt, text.c\_str());

    strcpy(pat, pattern.c\_str());

    KMPSearch(pat, txt);

    return 0;

}

**Output:**

Text

Description automatically generated