Name: Shaan Agarwal
UID: 2021300001
Title: Design and Analysis Of Algorithms Lab
Experiment No.: 9

Aim: To implement string matching algorithms using Rabin Karp.

Theory:

Rabin-Karp algorithm is an algorithm used for searching/matching patterns in the text using a hash function. Unlike Naive string matching algorithm, it does not travel through every character in the initial phase rather it filters the characters that do not match and then performs the comparison.

Algorithm:

```
n = t.length
m = p.length
h = dm-1 mod q
p = 0
t0 = 0
for i = 1 to m
    p = (dp + p[i]) mod q
    t0 = (dt0 + t[i]) mod q
for s = 0 to n - m
    if p = ts
        if p[1.....m] = t[s + 1..... s + m]
            print "pattern found at position" s
    If s < n-m
```

$$ts + 1 = (d (ts - t[s + 1]h) + t[s + m + 1]) \bmod q$$

Program

1. Naïve Method

```cpp
#include <iostream>
#include <string>

using namespace std;

int main() {
    string text, pattern;
    cout << "Enter the text string: ";
    getline(cin, text);
    cout << "Enter the pattern string: ";
    getline(cin, pattern);

    int n = text.length();
    int m = pattern.length();
    int count = 0;

    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 << "Pattern found at index " << i << endl;
            count++;
        }
    }

    if (count == 0) {
        cout << "Pattern not found in text." << endl;
    } else {
        cout << "Pattern found " << count << " times in text." << endl;
    }
```

```
    return 0;
}
```

Output

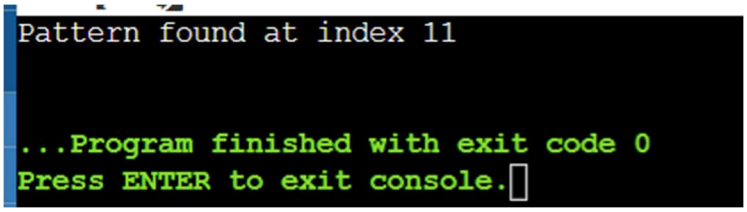
```
Enter the text string: MY NAME IS SHAAN
Enter the pattern string: SHAAN
Pattern found at index 11
Pattern found 1 times in text.


...Program finished with exit code 0
Press ENTER to exit console.
```

2) Rabin-Karp Method

```
#include <iostream>
#include <string>

using namespace std;

const int prime = 13;

int rabin_karp(string text, string pattern) {
    int n = text.size();
    int m = pattern.size();
    int pattern_hash = 0;
    int text_hash = 0;
    int h = 1;

    for (int i = 0; i < m - 1; i++) {
        h = (h * 256) % prime;
    }

    for (int i = 0; i < m; i++) {
        pattern_hash = (256 * pattern_hash + pattern[i]) % prime;
        text_hash = (256 * text_hash + text[i]) % prime;
    }

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

```
        if (pattern_hash == text_hash) {
            int j;
            for (j = 0; j < m; j++) {
                if (text[i + j] != pattern[j]) {
                    break;
                }
            }

            if (j == m) {
                return i;
            }
        }

        if (i < n - m) {
            text_hash = (256 * (text_hash - text[i] * h) + text[i + m]) % prime;

            if (text_hash < 0) {
                text_hash += prime;
            }
        }
    }

    return -1;
}

int main() {
    string text = "MY NAME IS SHAAN";
    string pattern = "SHAAN";
    int result = rabin_karp(text, pattern);

    if (result == -1) {
        cout << "Pattern not found in text" << endl;
    } else {
        cout << "Pattern found at index " << result << endl;
    }

    return 0;
}
```

Output



Conclusion: After performing the above experiment, I have understood the implementation of String Matching, and its implementation using the Rabin Karp Algorithm.