	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	**AIM: String Matching Rabin-Karp Approach**	
Experiment No: 26	**Date: 31/10/2023**	**Enrolment No: 92100133020**

String Matching Rabin-Karp Approach:

The Rabin-Karp algorithm uses hashing to find a pattern in a text. It hashes the pattern and compares it with the hashes of all substrings of the text.

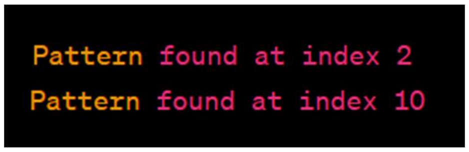
Algorithm:

1. Compute the hash value of the pattern and the first **m** characters of the text.
2. Iterate through the text from 1 to **n - m + 1**: a. If the hash value of the current substring matches the pattern's hash value, check character by character. b. Recompute the hash value for the next substring.

Code:

```
#include <iostream>
#include <string>
using namespace std;

const int prime = 101; // A prime number

int calculateHash(string str, int len) {
  int hash = 0;
  for (int i = 0; i < len; i++) {
    hash += (int)str[i] * pow(prime, i);
  }
  return hash;
}

void rabinKarpStringMatch(string text, string pattern) {
  int n = text.length();
  int m = pattern.length();
  int patternHash = calculateHash(pattern, m);
  int textHash = calculateHash(text, m);

  for (int i = 1; i <= n - m + 1; i++) {
    if (patternHash == textHash) {
      int j;
      for (j = 0; j < m; j++) {
        if (text[i + j - 1] != pattern[j])
          break;
      }
      if (j == m)
        cout << "Pattern found at index " << i - 1 << endl;
    }
    if (i < n - m + 1) {
            textHash = (textHash - text[i - 1]) / prime + text[i + m - 1] * pow(prime, m - 1);
        }
    }
}

int main() {
    string text = "ababcabcabababcabc";
    string pattern = "abc";
    rabinKarpStringMatch(text, pattern);
    return 0;
}
```

Output:



```
Pattern found at index 2
Pattern found at index 10
```

Space complexity: __________

Justification:____
____

Time complexity:

Best case time complexity: __________

Justification:____
____

Worst case time complexity: __________

Justification:____