**UCID: 2024510001 BATCH: A NAME: Atharva Vasant Angre**

**EXPERIMENT NO:   10**

**EXPERIMENT TITLE:** To implement String matching algorithm (Rabin Karp Algorithm)

**Objective:**

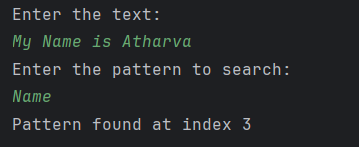
1.To understand how to find a pattern in a text using the Rabin-Karp algorithm.

2.To learn how hashing helps match strings quickly.

**Program code:** -

import java.util.Scanner;  
  
public class RabinKarp {  
 static final int *d* = 256; // Number of characters in input alphabet  
  
 public static void rabinKarpSearch(String text, String pattern, int q) {  
 int m = pattern.length();  
 int n = text.length();  
 int i, j;  
 int p = 0; // hash value for pattern  
 int t = 0; // hash value for text  
 int h = 1;  
  
 // The value of h would be "pow(d, m-1)%q"  
 for (i = 0; i < m - 1; i++)  
 h = (h \* *d*) % q;  
  
 // Calculate the hash value of pattern and first window of text  
 for (i = 0; i < m; i++) {  
 p = (*d* \* p + pattern.charAt(i)) % q;  
 t = (*d* \* t + text.charAt(i)) % q;  
 }  
  
 // Slide the pattern over text one by one  
 for (i = 0; i <= n - m; i++) {  
 // Check the hash values of current window and pattern  
 if (p == t) {  
 // If hash values match then only check characters one by one  
 for (j = 0; j < m; j++) {  
 if (text.charAt(i + j) != pattern.charAt(j))  
 break;  
 }  
  
 if (j == m)  
 System.*out*.println("Pattern found at index " + i);  
 }  
  
 // Calculate hash value for next window  
 if (i < n - m) {  
 t = (*d* \* (t - text.charAt(i) \* h) + text.charAt(i + m)) % q;  
  
 // Make sure that t >= 0  
 if (t < 0)  
 t = (t + q);  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.println("Enter the text: ");  
 String text = scanner.nextLine();  
  
 System.*out*.println("Enter the pattern to search: ");  
 String pattern = scanner.nextLine();  
  
 int q = 101; // A prime number  
  
 *rabinKarpSearch*(text, pattern, q);  
 }  
}

**Output:**

****

**Conclusion:**

In this experiment, we successfully implemented the Rabin-Karp algorithm to find a pattern within a given text using hashing. The algorithm efficiently calculates hash values to reduce unnecessary character comparisons, significantly speeding up the string matching process for large texts. This experiment enhanced our understanding of how hashing can be used in pattern matching and demonstrated the importance of collision handling and modular arithmetic in string algorithms.