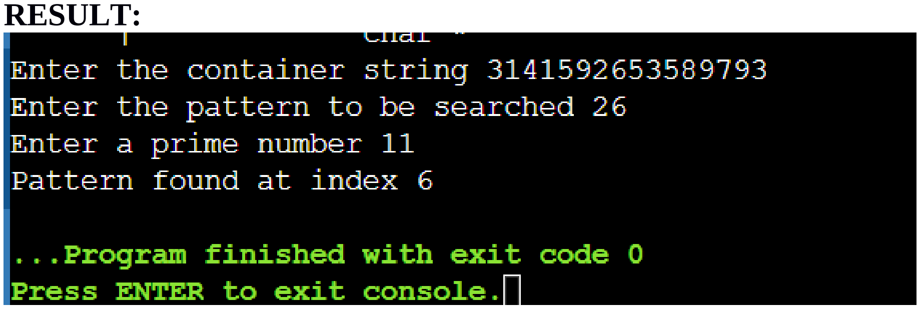
Name	Kinnari Shah
UID no.	2021700058
Experiment No.	10

AIM:	String Matching algorithm
Program 1	
PROBLEM STATEMENT :	To implement Rabin Karp’s Algorithm
ALGORITHM:	The Rabin-Karp string matching algorithm calculates a hash value for the pattern, as well as for each M-character subsequences of text to be compared. If the hash values are unequal, the algorithm will determine the hash value for next M-character sequence. If the hash values are equal, the algorithm will analyze the pattern and the M-character sequence. In this way, there is only one comparison per text subsequence, and character matching is only required when the hash values match. • Initially calculate the hash value of the pattern. • Start iterating from the starting of the string: • Calculate the hash value of the current substring having length m. • If the hash value of the current substring and the pattern are same check if the substring is same as the pattern. • If they are same, store the starting index as a valid answer. Otherwise, continue for the next substrings. • Return the starting indices as the required answer.

PROGRAM:

```
#include<stdio.h>
#include<string.h>

int main (){
   char txt[80], pat[80];
   int q;
   printf ("Enter the container string ");
   scanf ("%s", &txt);

   printf ("Enter the pattern to be searched ");
   scanf ("%s", &pat);

   int d = 256;
   printf ("Enter a prime number ");
   scanf ("%d", &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){
         for (j = 0; j < M; j++){
            if (txt[i + j] != pat[j])
            break;
         }
         if (j == M)
            printf ("Pattern found at index %d ", i);
      }
      if (i < N - M){
            t = (d * (t - txt[i] * h) + txt[i + M]) % q;
            if (t < 0)
              t = (t + q);
        }
    }
    return 0;
}
```

RESULT:



```
Enter the container string 3141592653589793
Enter the pattern to be searched 26
Enter a prime number 11
Pattern found at index 6

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

OBSERVATION:	Time complexity: The running time in the worst case scenario O ((n-m+1) *m). Average and best case running time is O (m+n) where m is length of pattern and n is length of text. Space complexity: **O(1)** We have used constant space. So, the space complexity is O(1).
CONCLUSION:	In this experiment, we implemented Rabin Karp’s Algorithm to find the matching string.