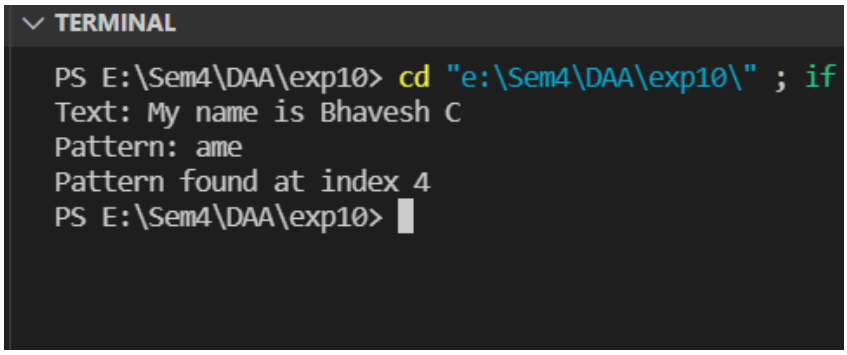
NAME:	Bhavesh Prashant Chaudhari
UID:	2021300018
SUBJECT	DAA
EXPERIMENT NO :	10
AIM:	To implement string matching algorithm using naive and Rabin Karp approach.
PROBLEM STATEMENT 1:	
ALGORITHM	There will be two loops: the outer loop will range from i=0 to i≤n-m, and the inner loop will range from j=0 to j<m, where 'm' is the length of the input pattern and n is the length of the text string. Now, At the time of searching algorithm in a window, there will be two cases: Case 1: If a match is found, we will match the entire pattern with the current window of the text string. And if found the pattern string is found in the current window after traversing, print the result. Else traverse in the next window. Case 2: If a match is not found, we will break from the loop(using the 'break' keyword), and the j pointer of the inner loop will move one index more and start the search algorithm in the next window
PROGRAM:	Naive- #include <stdio.h> #include <string.h>

```
void search(char* pat, char* txt)
{
    int M = strlen(pat);
    int N = strlen(txt);

    for (int i = 0; i <= N - M; i++) {
        int j;
        for (j = 0; j < M; j++)
            if (txt[i + j] != pat[j])
                break;
        if (j == M)
            printf("Pattern found at index %d \n", i);
    }
}

int main()
{
    char txt[] = "My name is Bhavesh C";
    char pat[] = "ame";
    printf("Text: %s\n",txt);
    printf("Pattern: %s\n",pat);
    search(pat, txt);
    return 0;
}
```

RESULT (SNAPSHOT)

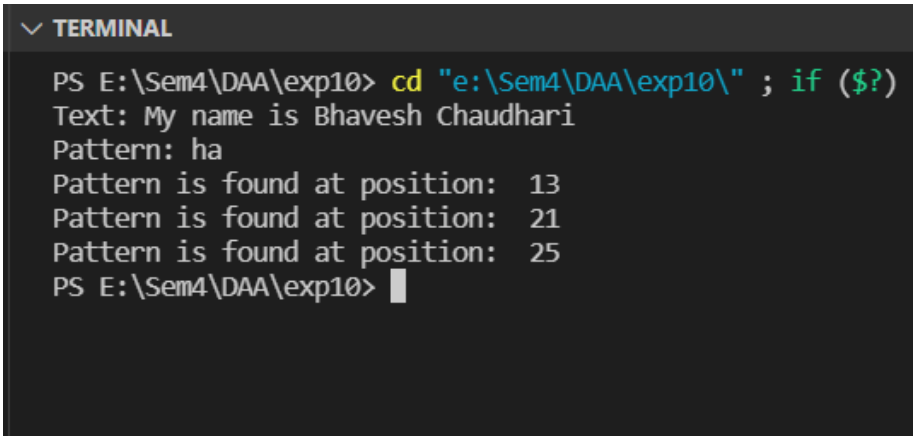


PROBLEM STATEMENT 2:	

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]) mod q

PROGRAM:	#include <stdio.h>
	#include <string.h>
	
	#define d 10
	
	void rabinKarp(char pattern[], char text[], int q) {
	  int m = strlen(pattern);
	  int n = strlen(text);
	  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 + pattern[i]) % q;
    t = (d * t + text[i]) % q;
  }

  for (i = 0; i <= n - m; i++) {
    if (p == t) {
      for (j = 0; j < m; j++) {
        if (text[i + j] != pattern[j])
          break;
      }

      if (j == m)
        printf("Pattern is found at position:  %d \n", i + 1);
    }

    if (i < n - m) {
      t = (d * (t - text[i] * h) + text[i + m]) % q;

      if (t < 0)
        t = (t + q);
    }
  }
}

int main() {
  char text[] = "My name is Bhavesh Chaudhari";
  char pattern[] = "ha";
  printf("Text: %s\n",text);
  printf("Pattern: %s\n",pattern);
  int q = 13;
  rabinKarp(pattern, text, q);
}
```

RESULT (SNAPSHOT)	
CONCLUSION:	Through this experiment I understood how to implement string matching algorithm using naive and rabinkarp approach.Naive approach has a O(n*m) time complexity while Rabin Karp has time complexity of O(m+n).