Algorithm Lab

Assignment-13

NAME SHISHU

REG: 2020CA089

1. Write a c program to implement string matching algorithm (Brute Force approach)

```c
#include <stdio.h>
  #include <string.h>
  #define MAX 100

  /* try to find the given pattern in the search string */
  int bruteForce(char *search, char *pattern, int slen, int plen) {
        int i, j, k;

        for (i = 0; i <= slen - plen; i++) {
                for (j = 0, k = i; (search[k] == pattern[j]) &&
                        (j < plen); j++, k++);
                if (j == plen)
                        return j;
        }
        return -1;
  }

  int main() {
        char searchStr[MAX], pattern[MAX];
        int res;
        printf("Enter Search String:");
        fgets(searchStr, MAX, stdin);
        printf("Enter Pattern String:");
        fgets(pattern, MAX, stdin);
        searchStr[strlen(searchStr) - 1] = '\0';
        pattern[strlen(pattern) - 1] = '\0';
        res = bruteForce(searchStr, pattern, strlen(searchStr), strlen(pattern));
        if (res == -1) {
                printf("Search pattern is not available\n");
        } else {
                printf("Search pattern available at the location %d\n", res);
        }
        return 0;
  }
```

```
Enter Search String:rahul
Enter Pattern String:ah
Search pattern available at the location 2
PS D:\cpp\sem3Algo\assignment13>
```

2. Write a c program to implement string matching algorithm (Rabin-Karp approach)

```c
#include<stdio.h>
#include<string.h>
int main (){
   char txt[80], pat[80];
   int q;
   printf ("Enter the container string \n");
   scanf ("%s", &txt);
   printf ("Enter the pattern to be searched \n");
   scanf ("%s", &pat);
   int d = 256;
   printf ("Enter a prime number \n");
   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 \n", i);
      }
      if (i < N - M){
         t = (d * (t - txt[i] * h) + txt[i + M]) % q;
         if (t < 0)
```

```
            t = (t + q);
        }
    }
    return 0;
}
```

```
Enter the container string
sldjflsjdfljslkfj
Enter the pattern to be searched
ldj
Enter a prime number
1
Pattern found at index 1
```