Q9. c) Implement Rabin Karp Algorithm. Analyze its time complexity.

TIME COMPLEXITY:
1. AVERAGE CASE: O(m + n)
2. BEST CASE: O(m + n)
3. WORST CASE: O(m * n)

SPACE COMPLEXITY: O(1)

where m is the length of pattern and n is the length of text.

```
#include <iostream>
#include <queue>
#include <vector>
#include <string>
#include <map>
#include <cmath>
#define d 256


using namespace std;


void search(string & pat, string & txt, int q)
{
    int M = int(pat.size());
    int N = int(txt.size());
    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)
                cout << "Pattern found at index: " << i << endl;
        }
        if(i < N - M)
        {
            t = (d * (t - txt[i] * h) + txt[i + M]) % q;
            if (t < 0)
                t = (t + q);
        }
```

```
    }
}


int main()
{
    string txt = "GEEKS FOR GEEKS";
    string pat = "GEEK";
    cout << "TEXT: " << txt << endl;
    cout << "PATTERN: " << txt << endl;
    int q = 37;
    search(pat, txt, q);
    return 0;
}
```

OUTPUT:

```
TEXT: GEEKS FOR GEEKS
PATTERN: GEEKS FOR GEEKS
Pattern found at index: 0
Pattern found at index: 10
Program ended with exit code: 0
```