RabinCarp in C++

```
#include <iostream>
#include <string>
using namespace std;

const int p = 31;
const int mod = 1e9 + 7;
long long poly_hash(const string& s) {
    long long hash = 0;
    long long p_power = 1;

    for (int i = 0; i < s.length(); i++) {
        hash = (hash + (s[i] - 'a' + 1) * p_power) % mod;
        p_power = (p_power * p) % mod;
    }

    return hash;
}
int powr(int a, int b) {
    // (a^b)%mod
    int res = 1;
    while (b > 0) {
        if (b & 1) res = (res * 1LL * a) % mod;
        a = (a * 1LL * a) % mod;
        b >>= 1;
    }
    return res;
}

int main() {
    string text = "ababbabbaba";
    string pattern = "aba";
    long long pat_hash = poly_hash(pattern);
    int n = text.length(), m = pattern.length();
    long long text_hash = poly_hash(text.substr(0, m));
    if (pat_hash == text_hash) {
        cout << 0 << endl;
    }
    for (int i = 1; i + m <= n; i++) {
        // remove last character
        text_hash = (text_hash - (text[i - 1] - 'a' + 1) +
mod) % mod;

        text_hash = (text_hash * 1LL * powr(p, mod - 2))
% mod;

        text_hash = (text_hash + (text[i + m - 1] - 'a' + 1)
* 1LL * powr(p, m - 1)) % mod;

        if (text_hash == pat_hash) {
            cout << i << endl;
        }
    }
    return 0;
}
```

Input:

- **Text** = "ababbabbaba"
- **Pattern** = "aba"
- **p = 31**, **mod = 1e9 + 7**

📥 Step 1: Compute pattern hash

Pattern: "a" (1), "b" (2), "a" (1)
Hash formula:
hash = (1*p^0 + 2*p^1 + 1*p^2) % mod
= (1*1 + 2*31 + 1*961) = 1 + 62 + 961 = 1024

📥 Step 2: Slide over text & compare hash window

We'll use a table with:

Index i	Substring text[i..i+2]	Rolling Hash	Matches pat_hash = 1024?
0	a b a	1024	✅ Yes
1	b a b	2973	❌ No
2	a b b	2086	❌ No
3	b b a	2853	❌ No
4	b a b	2973	❌ No
5	a b b	2086	❌ No
6	b b a	2853	❌ No
7	b a b	2973	❌ No
8	a b a	1024	✅ Yes

✅ Matches found at indices:

0
8

```
0
8
```