```cpp
//
//   Find the number of integers between 1 and K (inclusive)
//   satisfying the following condition, modulo 10^9 + 7:
//     - The sum of the digits in base ten is a multiple of D.
//
//     Time Complexity: O(ND)
//

#include <bits/stdc++.h>
#define ll long long

using namespace std;

const int MOD = 1e9 + 7;
inline void uadd(int& a, int b) {
    a += b;
    if (a >= MOD) a -= MOD;
}

int main() {
    string s; cin >> s;
    int v; cin >> v;
    int n = s.size();

    if (n == 1) {
        int ans = 0;
        for (int i = 1; i <= (s[0] - '0'); i++) {
            if (i % v == 0) ans++;
        }
        cout << ans << endl;
        return 0;
    }

    vector<int> pref(n);
    pref[0] = (s[0] - '0') % v;
    for (int i = 1; i < n; i++) {
        pref[i] = (pref[i-1] + s[i] - '0') % v;
    }

    // backwards dynamic programming
    vector<vector<int>> bw(n, vector<int>(v, 0));
    int ans = 0;

    // Base case
    for (int d = 0; d < 10; d++) {
        bw[n-1][d%v] += 1;
        if (d <= s[n-1] - '0' && n-2 >= 0 && (pref[n-2] + d) % v == 0) ans++;
    }

    for (int i = n - 2; i >= 0; i--) {
        for (int d = 0; d <= (i == 0 ? s[0] - '0' - 1 : 9); d++) {
            for (int j = 0; j < v; j++) {
                uadd(bw[i][(j+d)%v], bw[i+1][j]);
            }
        }

        if (i-1 >= 0) {
            for (int d = 0; d < s[i] - '0'; d++) {
                uadd(ans, bw[i+1][(v - ((pref[i-1] + d + 100 * v) % v)) % v]);
            }
        }
    }

    cout << ((ll)ans + bw[0][0] - 1 + MOD) % MOD << endl;
    return 0;
}
```