Recursive Digit Sum

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'superDigit' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. STRING n
 *  2. INTEGER k
 */

int superDigit(string n, int k) {
    long long sum=0;
    // int m=n.size();
    for(char c : n){
        sum+=(c-'0');
    }

    sum *= k;
    while(sum>=10){
        long long num=0;
        while (sum > 0) {
            num += sum % 10;
            sum /= 10;
        }
        sum=num;
    }
    return sum;
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string first_multiple_input_temp;
    getline(cin, first_multiple_input_temp);
```

```
    vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));

    string n = first_multiple_input[0];

    int k = stoi(first_multiple_input[1]);

    int result = superDigit(n, k);

    fout << result << "\n";

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int,
int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;
```

```
    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}
```