PREPARE NEW CERTIFY COMPETE Search 210554M_CSE_21

All Contests > In21-CS2023-Lab4 > Recursive Digit Sum

Recursive Digit Sum

Problem Submissions Leaderboard Discussions

We define super digit of an integer x using the following rules:

Given an integer, we need to find the *super digit* of the integer.

- If x has only 1 digit, then its super digit is x.
- Otherwise, the super digit of x is equal to the super digit of the sum of the digits of x.

For example, the super digit of 9875 will be calculated as:

```
	super_digit(9875)   	9+8+7+5 = 29 
	super_digit(29) 	2 + 9 = 11
	super_digit(11)		1 + 1 = 2
	super_digit(2)		= 2  
```

Example

$n =' 9875'$
$k = 4$

The number p is created by concatenating the string n k times so the initial $p = 9875987598759875$.

```
    superDigit(p) = superDigit(9875987598759875)
                  9+8+7+5+9+8+7+5+9+8+7+5+9+8+7+5 = 116
    superDigit(p) = superDigit(116)
                  1+1+6 = 8
    superDigit(p) = superDigit(8)
```

All of the digits of p sum to 116. The digits of 116 sum to 8. 8 is only one digit, so it is the super digit.

Function Description

Complete the function *superDigit* in the editor below. It must return the calculated super digit as an integer.

superDigit has the following parameter(s):

- *string n:* a string representation of an integer
- *int k:* the times to concatenate n to make p

Returns

- *int:* the super digit of n repeated k times

Input Format

The first line contains two space separated integers, n and k.

Constraints

- $1 \leq n < 10^{100000}$
- $1 \leq k \leq 10^5$

Sample Input 0

```
148 3
```

Sample Output 0

```
3
```

Explanation 0

Here $n = 148$ and $k = 3$, so $p = 148148148$.

```
super_digit(P) = super_digit(148148148)
               = super_digit(1+4+8+1+4+8+1+4+8)
               = super_digit(39)
               = super_digit(3+9)
               = super_digit(12)
               = super_digit(1+2)
               = super_digit(3)
               = 3
```

Sample Input 1

```
9875 4
```

Sample Output 1

```
8
```

Sample Input 2

```
123 3
```

Sample Output 2

```
9
```

Explanation 2

Here $n = 123$ and $k = 3$, so $p = 123123123$.

```
super_digit(P) = super_digit(123123123)
               = super_digit(1+2+3+1+2+3+1+2+3)
               = super_digit(18)
               = super_digit(1+8)
               = super_digit(9)
               = 9
```

Contest ends in 10 hours

Submissions: 116
Max Score: 50
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

C++

```cpp
#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) {

}

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;
}
```

```cpp

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;
}

```

Line: 1 Col: 1

Upload Code as File □ Test against custom input Run Code Submit Code

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |