Started on	Tuesday, 13 February 2024, 6:39 PM
State	Finished
Completed on	Tuesday, 13 February 2024, 7:34 PM
Time taken	55 mins 21 secs
Marks	20.00/20.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 10.00 out of 10.00

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
```

For example:

Input	Result
148 3	3
9875 4	8
123 3	9

Answer: (penalty regime: 0 %)

Reset answer

```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) {
    // Base case: if n has only one digit
    if (n.length() == 1) {
        return stoi(n); // Convert single-digit string to integer and return
    }

    // Calculate the digit sum of the string n
    long long digitSum = 0; // Using long long to handle large numbers
    for (char c : n) {
        digitSum += (c - '0'); // Convert char to integer and add to digitSum
    }

    // digit sum by * k
    digitSum *= k;

    // Recursive superDigit with the new string representation of digitSum
    return superDigit(to_string(digitSum), 1);

}

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);

```

	Input	Expected	Got	
✓	148 3	3	3	✓
✓	9875 4	8	8	✓
✓	123 3	9	9	✓

Passed all tests! ✓

‣ **Show/hide question author's solution (Cpp)**

Correct

Marks for this submission: 10.00/10.00.

Question 2

Correct

Mark 10.00 out of 10.00

Find the number of ways that a given integer, X, can be expressed as the sum of the N^{th} powers of unique, natural numbers.

For example, if $X = 13$ and $N = 2$, we have to find all combinations of unique squares adding up to 13. The only solution is $2^2 + 3^2$.

Function Description

Complete the *powerSum* function in the editor below. It should return an integer that represents the number of possible combinations.

powerSum has the following parameter(s):

- *X*: the integer to sum to
- *N*: the integer power to raise numbers to

Input Format

The first line contains an integer X.
The second line contains an integer N.

Constraints

- $1 \leq X \leq 1000$
- $2 \leq N \leq 10$

Output Format

Output a single integer, the number of possible combinations caclulated.

Sample Input 0

```
10
2
```

Sample Output 0

```
1
```

Explanation 0

If $X = 10$ and $N = 2$, we need to find the number of ways that 10 can be represented as the sum of squares of unique numbers.

$10 = 1^2 + 3^2$

This is the only way in which 10 can be expressed as the sum of unique squares.

Sample Input 1

```
100
2
```

Sample Output 1

```
3
```

Explanation 1

$100 = (10^2) = (6^2 + 8^2) = (1^2 + 3^2 + 4^2 + 5^2 + 7^2)$

Sample Input 2

```
100
3
```

Sample Output 2

```
1
```

Explanation 2

100 can be expressed as the sum of the cubes of $1, 2, 3, 4$.
$(1 + 8 + 27 + 64 = 100)$. There is no other way to express 100 as the sum of cubes.

For example:

Input	Result
10 2	1

Input	Result
100 2	3
100 3	1

Answer: (penalty regime: 0 %)

Reset answer

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

using namespace std;

string ltrim(const string &);
string rtrim(const string &);

/*
 * Complete the 'powerSum' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER X
 *  2. INTEGER N
 */
//int num =1;
int powerSum(int X, int N, int num=1) {
    // Calculate the value after subtracting the Nth power of the current number from X
    int val = X - pow(num, N);//100-1=99

    // If the value becomes negative, return 0 (no valid combination)
    if (val < 0) {
        return 0;
    }
    // If the value becomes 0, return 1 (a valid combination is found)
    else if (val == 0) {
        return 1;
    }
    // Otherwise, recursively call powerSum with updated X and the next natural number
    else {
        return powerSum(val, N, num + 1) + powerSum(X, N, num + 1);//(1,2,2)+(100,2,2)
    }
}

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

    string X_temp;
    getline(cin, X_temp);

    int X = stoi(ltrim(rtrim(X_temp)));

    string N_temp;
    getline(cin, N_temp);

    int N = stoi(ltrim(rtrim(N_temp)));

    int result = powerSum(X, N);
    cout << result << "\n";
    fout << result << "\n";

```

	Input	Expected	Got	
✓	10 2	1	1	✓
✓	100 2	3	3	✓
✓	100 3	1	1	✓

Passed all tests! ✔

▸ **Show/hide question author's solution (Cpp)**

Correct

Marks for this submission: 10.00/10.00.