

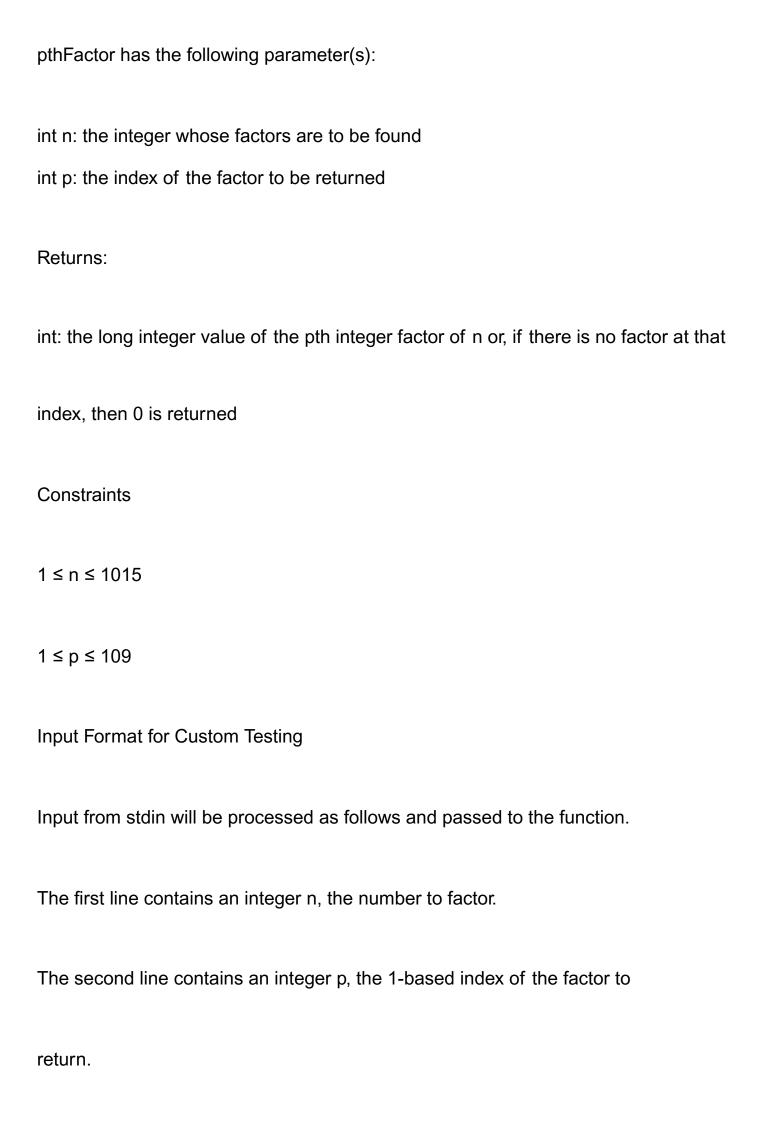
```
1 + /*
     * Complete the 'fourthBit' function below.
 2
 3
     * The function is expected to return an INTEGER.
 4
 5
     * The function accepts INTEGER number as parameter.
 6
 7
    int fourthBit(int number)
 8
 9 + {
10
        int binary[32];
11
        int i = 0;
12
        while(number > 0)
13 ,
            binary[i] = number % 2;
14
15
            number /= 2;
16
            1++;
17
        if(i >= 4)
18
19 ,
        {
20
           return binary[3];
21
        else
22
23
        return 0;
   }
24
```

	Test	Expected	Got	
~	printf("%d", fourthBit(32))	0	0	~
~	printf("%d", fourthBit(77))	1	1	~

Passed all tests! <

Problem Statement:2
Determine the factors of a number (i.e., all positive integer values that evenly divide into
a number) and then return the pth element of the list, sorted ascending. If
there is no pth
element, return 0.
Example
n = 20
p = 3
The factors of 20 in ascending order are $\{1, 2, 4, 5, 10, 20\}$ . Using 1-based indexing, if $p =$
3, then 4 is returned. If p > 6, 0 would be returned.
Function Description

Complete the function pthFactor in the editor below.



Sample Input

STDIN Function

• -----

$$10 \rightarrow n = 10$$

 $3 \rightarrow p = 3$  Sample Output 5 Explanation

Factoring n = 10 results in  $\{1, 2, 5, 10\}$ . Return the p = 3rd factor, 5, as the answer.

•

```
* Complete the 'pthFactor' function below.
 2
 3
    * The function is expected to return a LONG_INTEGER.
 4
    * The function accepts following parameters:
 5
    * 1. LONG INTEGER n
 6
    * 2. LONG_INTEGER p
 7
    */
 8
 9
10 long pthFactor(long n, long p)
11 + {
        int count = 0;
12
        for(long i = 1; i <= n; ++i)
13
14 +
            if(n % i == 0)
15
16 +
17
                count++;
               if(count == p)
18
19 v
                    return i;
20
                }
21
22
23
24
        return 0;
25 }
```

	Test	Expected	Got	
~	printf("%ld", pthFactor(10, 3))	5	5	~
~	printf("%ld", pthFactor(10, 5))	ө	0	~
~	printf("%ld", pthFactor(1, 1))	1	1	~

Passed all tests! <

Problem Statement:3
You are a bank account hacker. Initially you have 1 rupee in your account, and you want
exactly N rupees in your account. You wrote two hacks, first hack can multiply the amount
of money you own by 10, while the second can multiply it by 20. These hacks can be used
any number of time. Can you achieve the desired amount N using these hacks.
Constraints:



2

## SAMPLE OUTPUT

0

```
* Complete the 'myFunc' function below.
3
    * The function is expected to return an INTEGER.
4
    * The function accepts INTEGER n as parameter.
5
    */
6
   int myFunc(int n)
8
9 + {
        if(n == 1) return 1;
10
        if(n % 10 == 0 && myFunc(n / 10)) return 1;
11
12
        if(n % 20 == 0 && myFunc(n / 20)) return 1;
13
        return 0;
   }
14
15
```

	Test	Expected	Got	
~	printf("%d", myFunc(1))	1	1	~
~	printf("%d", myFunc(2))	0	0	~
~	printf("%d", myFunc(10))	1	1	~
~	printf("%d", myFunc(25))	0	0	~
~	printf("%d", myFunc(200))	1	1	~

## **Problem Statement:4**

Find the number of ways that a given integer, X, can be expressed as the sum of the Nth

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 22 + 32.

**Function Description** 

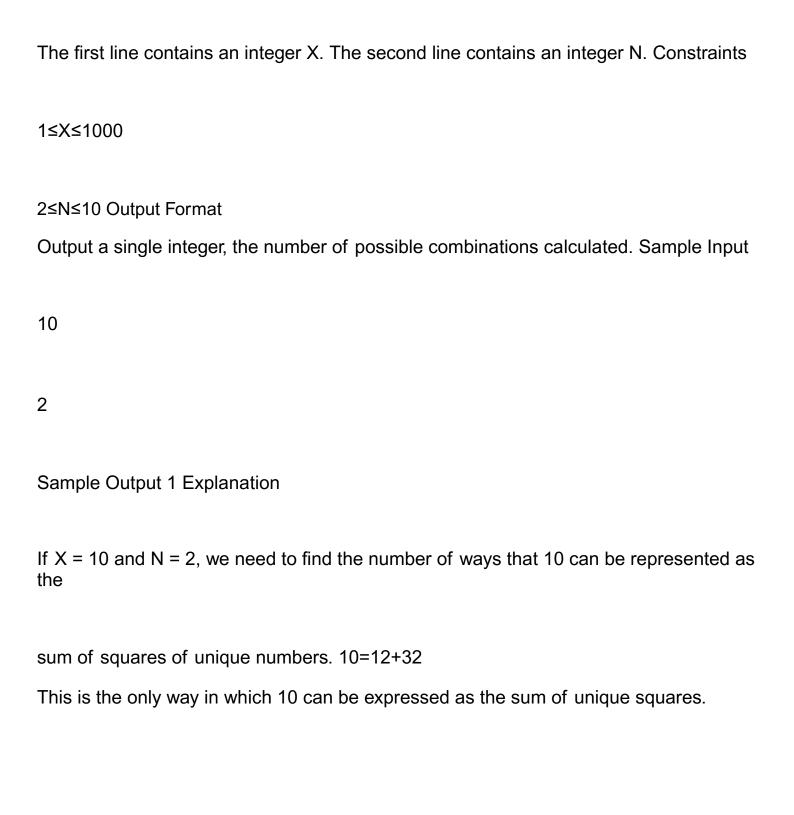
Complete the powerSum function in the editor below. It should return an integer

that represents the number of possible combinations.

powerSum has the following parameter(s):

X: the integer to sum to

• the integer power to raise numbers to Input Format



	Test	Expected	Got	
~	printf("%d", powerSum(10, 1, 2))	1	1	~
	printr( au , power-sum(10, 1, 2/)	1	1	_

Finish review