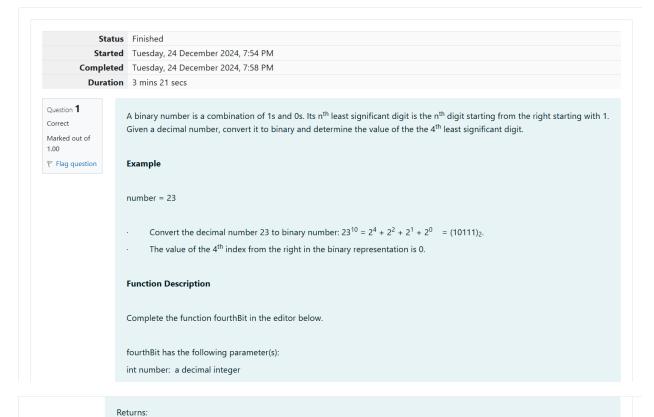
Week-12-User-Defined Functions

Coding

ROLL NO: 241801294

NAME: THARUN N

Q1)



0

Explanation 0

- · Convert the decimal number 32 to binary number: $32_{10} = (100000)_2$.
- \cdot The value of the 4th index from the right in the binary representation is 0.

Sample Case 1

Sample Input 1

```
STDIN Function
-----
77 → number = 77
```

Sample Output 1

.

Explanation 1

- · Convert the decimal number 77 to binary number: $77_{10} = (1001101)_2$.
- The value of the 4th index from the right in the binary representation is 1.

	Test	Expected	Got	
~	<pre>printf("%d", fourthBit(32))</pre>	0	0	~
~	printf("%d", fourthBit(77))	1	1	~
Passed all tests! ✓				



Question **2**Correct
Marked out of 1.00
F Flag question

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the p^{th} element of the list, sorted ascending. If there is no p^{th} 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.

pthFactor has the following parameter(s):

int n: the integer whose factors are to be found

int p: the index of the factor to be returned

Returns:

int: the long integer value of the p^{th} integer factor of n or, if there is no factor at that index, then 0 is returned

Constraints

 $1 \le n \le 10^{15}$

 $1 \le p \le 10^9$

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n, the number to factor.

The second line contains an integer p, the 1-based index of the factor to return.

Sample Case 0

Sample Input 0

STDIN Function

10 → n = 10

 $3 \rightarrow p = 3$

Sample Output 0

5

Explanation 0

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

Sample Case 1

Sample Input 1

STDIN Function
----10 → n = 10

Sample Output 1

 $5 \rightarrow p = 5$

0

Explanation 1

Factoring n = 10 results in $\{1, 2, 5, 10\}$. There are only 4 factors and p = 5, therefore 0 is returned as the answer.

Sample Case 2

Sample Input 2

```
STDIN Function

1 → n = 1

1 → p = 1
```

Sample Output 2

1

Explanation 2

Factoring n = 1 results in {1}. The p = 1st factor of 1 is returned as the answer.

```
* The function is expected to return a LONG_INTEGER.
5 * The function accepts following parameters:
6 * 1. LONG_INTEGER n
7 * 2. LONG_INTEGER p
10 long pthFactor(long n, long p)
11 🔻 {
         long j = 0;
for (long i = 1; i <= n; i++) {
    if (n%i == 0) {
        j++;
12
13 •
14 •
           j++;
}
15
16
17
         if ( j == p ) {
    return i;
}
18 •
19
20
21
22
          return 0;
23
24 }
```

	Test	Expected	Got	
~	<pre>printf("%ld", pthFactor(10, 3))</pre>	5	5	~
~	<pre>printf("%ld", pthFactor(10, 5))</pre>	0	0	~
~	<pre>printf("%ld", pthFactor(1, 1))</pre>	1	1	~

Passed all tests! ✓

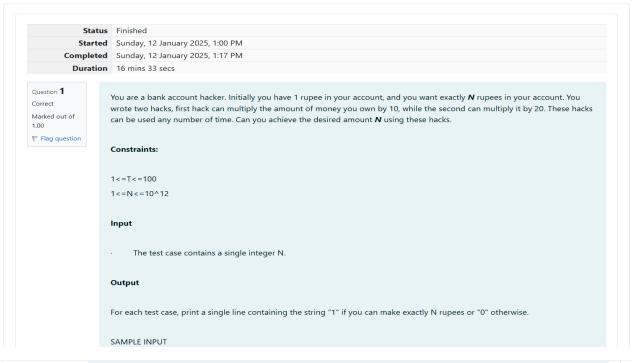
Week-12-Recursive Functions

Coding

ROLL NO: 241801294

NAME: THARUN N

Q1)



```
* Complete the 'myFunc' function below.
 3
     \boldsymbol{*} The function is expected to return an <code>INTEGER.</code>
 4
     \ensuremath{^{*}} The function accepts INTEGER n as parameter.
 5
 8 int myFunc(int n)
9 + {
10 •
         if ((n\%10 == 0) || ((n\%20) == 0) || n == 1) {
11
            return 1;
12
13
         return 0;
14
15 }
16
```

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

Passed all tests! ✓

Q2)

Question **2**Correct
Marked out of 1.00

Flag question

Find the number of ways that a given integer, \mathbf{X} , can be expressed as the sum of the \mathbf{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 \boldsymbol{X} .

The second line contains an integer N.

Constraints

 $1 \le X \le 1000$

 $2 \le N \le 10$

Output a single integer, the number of possible combinations calculated.			
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			
100			
100 2			
100 2 Sample Output 1			
100 2 Sample Output 1			
2 Sample Output 1 3 Explanation 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 Output 1 Sample Output 1 $100 = (10^2) = (6^2 + 8^2) = (1^2 + 3^2 + 4^2 + 5^2 + 7^2)$ Sample Input 2			

Output Format

Sample Output 2

Explanation 2

100 can be expressed as the sum of the cubes of 1, 2, 3, 4.

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.

Answer: (penalty regime: 0 %)

```
Reset answer
```

```
* Complete the 'powerSum' function below.
 3
    * The function is expected to return an INTEGER.

* The function accepts following parameters:
4
5
     * 1. INTEGER x
 6
     * 2. INTEGER n
10 #include <math.h>
11
12 int powerSum(int x, int m, int n)
13 v {
14 🔻
        if (x == 0) {
15
           return 1;
16
17
18
       if (x < 0) {
20
22
        int c = 0;
23
24
        for (int i = m; i*i <= x; i++) {
25
           int p = pow(i, n);
26
27
            c+=powerSum(x-p, i+1, n);
28
```

```
23
24 v
for (int i = m; i*i <= x; i++) {
    int p = pow(i, n);
    c+=powerSum(x-p, i+1, n);
    28
29    }
30
31    return c;
32 }
```

	Test	Expected	Got	
~	<pre>printf("%d", powerSum(10, 1, 2))</pre>	1	1	~

Passed all tests! 🗸