



Week-12-User defined and recursive functions

Question 1

Correct

Marked out of
1.00[Flag question](#)

A binary number is a combination of 1s and 0s. Its n^{th} least significant digit is the n^{th} digit starting from the right starting with 1. Given a decimal number, convert it to binary and determine the value of the the 4^{th} least significant digit.

Example

```
number = 23
```

- Convert the decimal number 23 to binary number: $23^{10} = 2^4 + 2^2 + 2^1 + 2^0 = (10111)_2$.
- The value of the 4^{th} index from the right in the binary representation is 0.

Function Description

Complete the function fourthBit in the editor below.

fourthBit has the following parameter(s):

int number: a decimal integer

Returns:

int: an integer 0 or 1 matching the 4th least significant digit in the binary representation of number.

Constraints

$0 \leq \text{number} < 2^{31}$

Input Format for Custom Testing

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

The only line contains an integer, `number`.

Sample Case 0**Sample Input 0**

STDIN Function

32 → `number = 32`

Sample Output 0

0

Explanation 0

- Convert the decimal number 32 to binary number: $32_{10} = (100000)_2$.

- 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

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.

Source code

```
7
8 int fourthBit(int number)
9 {
10    int binary[32];
11    int i=0;
12    while(number>0){
13        binary[i]=number%2;
14        number/=2;
15        i++;
16    }
17    if(i>=4){
18        return binary[3];
19    }
20    else
21        return 0;
22 }
```

Result

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

Passed all tests! ✓

Question 2

Correct

Marked out of
1.00

[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

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

$1 \leq p \leq 10^9$

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   →  p = 3
```

Sample Output 0

5

Explanation 0

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

Sample Case 1**Sample Input 1**

STDIN Function

```
----  
10  →  n = 10  
5   →  p = 5
```

Sample Output 1

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.

Source code

```
8  */  
9  
10 long pthFactor(long n, long p)  
11 {  
12     int count=0;  
13     for(long i=1;i<=n;++i){  
14         if(n%i==0){  
15             count++;  
16             if(count==p){  
17                 return i;  
18             }  
19         }  
20     }  
21     return 0;  
22 }
```

Result

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

Passed all tests! ✓

Question 1

Correct

Marked out of
1.00[Flag question](#)

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: $1 \leq T \leq 100$ $1 \leq N \leq 10^{12}$ **Input**

- The test case contains a single integer N .

Output

For each test case, print a single line containing the string "1" if you can make exactly N rupees or "0" otherwise.

SAMPLE INPUT

1

SAMPLE OUTPUT

1

SAMPLE INPUT

2

SAMPLE OUTPUT

0

Source code

```
1 /*
2  * Complete the 'myFunc' function below.
3  *
4  * The function is expected to return an INTEGER.
5  * The function accepts INTEGER n as parameter.
6  */
7
8 int myFunc(int n)
9 {
10     return n%10==0 || n==1;
11 }
```

Result

	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	✓

Passed all tests! ✓

Question 2

Correct

Marked out of
1.00[Flag question](#)

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

$1 \leq X \leq 1000$

Output Format

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  
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.

Source code

```
o   //  
9  
10 int powerSum(int x, int m, int n)  
11 {  
12     int power=m;  
13     for(int i=1;i<n;i++){  
14         power*=m;  
15     }  
16     if(power>x){  
17         return 0;  
18     }  
19     if(power==x){  
20         return 1;  
21     }  
22     return powerSum(x-power,m+1,n)+powerSum(x,m+1,n);  
23 }
```

Result

	Test	Expected	Got	
✓	printf("%d", powerSum(10, 1, 2))	1	1	✓

Passed all tests! ✓