MUKUNDAN.D

240801210
ECE-D

Problem Statement:1
A binary number is a combination of 1s and 0s. Its nth least significant digit is the nth digit
starting from the right starting with 1. Given a decimal number, convert it to binary and
determine the value of the the 4th least significant digit.
Example
number = 23
• Convert the decimal number 23 to binary number: 2310 = 24 + 22 + 21 + 20 =
(10111)2.
• The value of the 4th 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 ≤ number < 231
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 Input
STDIN Function
----- --------
32 → number = 32
Sample Output
0

Explanation

• Convert the decimal number 32 to binary number: 3210 = (100000)2.

• The value of the 4th index from the right in the binary representation is 0.

```c
/*
 * Complete the 'fourthBit' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER number as parameter.
 */

int fourthBit(int number)
{
    int binary[32];
    int i = 0;
    while(number > 0)
    {
        binary[i] = number % 2;
        number /= 2;
        i++;
    }
    if(i >= 4)
    {
        return binary[3];
    }
    else
    return 0;
}
```

	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.
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 pth integer factor of n or, if there is no factor at that
index, then 0 is returned
Constraints
$1 \le n \le 1015$
$1 \le p \le 109$
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 Input

```
STDIN Function
----- --------
10 → n = 10
3 → p = 3
```

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

```c
/*
 * Complete the 'pthFactor' function below.
 *
 * The function is expected to return a LONG_INTEGER.
 * The function accepts following parameters:
 *  1. LONG_INTEGER n
 *  2. LONG_INTEGER p
 */

long pthFactor(long n, long p)
{
    int count = 0;
    for(long i = 1; i <= n; ++i)
    {
        if(n % i == 0)
        {
            count++;
            if(count == p)
            {
                return i;
            }
        }
    }
    return 0;
}
```

	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! ✓

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:
1<=T<=100
1<=N<=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

```c
 * Complete the 'myFunc' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER n as parameter.
 */

int myFunc(int n)
{
    if(n == 1) return 1;
    if(n % 10 == 0 && myFunc(n / 10)) return 1;
    if(n % 20 == 0 && myFunc(n / 20)) return 1;
    return 0;
}

```

	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! ✓

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
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 ≤ X ≤ 1000
2 ≤ N ≤ 10
Output Format
Output a single integer, the number of possible combinations calculated.
Sample Input
10
2
Sample Output
1
Explanation
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 = 12 + 32

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

```c
/*
 * 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 powerSum(int x, int m, int n)
{
    int power = 1;
    for(int i = 0; i < n; i++)
    power *= m;
    if(power > x) return 0;
    if(power == x) return 1;
    return powerSum(x - power, m + 1, n) + powerSum(x, m + 1, n);
}
```

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

Passed all tests! ✓

Finish review