Programming using C

week 12 practice session and coding

Name:**S.Jananejaipratha**
Department:**AIML-'B'**
Roll No.:**241501079**

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 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 < 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
```

Reset answer

```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! ✓

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 $p = 3^{rd}$ factor, 5, as the answer.

Sample Case 1
Sample Input 1

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

Sample Output 1

```
0
```

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 {}. The p = 1st factor of 1 is returned as the answer.

Answer: (penalty regime: 0 %)

Reset answer

```c
/*
 * Complete the 'ptFactor' 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 ptFactor(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", ptFactor(10, 3))	5	5	✓
✓	printf("%ld", ptFactor(10, 9))	0	0	✓
✓	printf("%ld", ptFactor(1, 1))	1	1	✓

Passed all tests! ✓