

GE23131-Programming Using C-2024

Quiz navigation

1

2

3

Show one page at a time

Finish review

Status	Finished
Started	Wednesday, 25 December 2024, 10:16 PM
Completed	Wednesday, 25 December 2024, 10:19 PM
Duration	3 mins 15 secs

Question 1

Correct

Marked out of 1.00

Flag question

Objective

In this challenge, we're going to use loops to help us do some simple math. Check out the Tutorial tab to learn more.

Task

Given an integer, n , print its first 10 multiples. Each multiple $n \times i$ (where $1 \leq i \leq 10$) should be printed on a new line in the form: $n \times i = \text{result}$.

Input Format

A single integer, n .

Constraints

$2 \leq n \leq 20$

Output Format

Print 10 lines of output; each line i (where $1 \leq i \leq 10$) contains the **result** of $n \times i$ in the form:

$n \times i = \text{result}$.

Sample Input

2

Sample Output

$2 \times 1 = 2$
 $2 \times 2 = 4$
 $2 \times 3 = 6$
 $2 \times 4 = 8$
 $2 \times 5 = 10$
 $2 \times 6 = 12$
 $2 \times 7 = 14$
 $2 \times 8 = 16$
 $2 \times 9 = 18$
 $2 \times 10 = 20$

Answer: (penalty regime: 0 %)

1

2

3

4

5

6

7

8

9

10

#include<stdio.h>

int main()

{

int n;

scanf("%d",&n);

```

11  for(int i=1;i<=10;i++){
12
13  printf("%d x %d = %d\n",n,i,i*n);
14
15  }
16
17  }

```

	Input	Expected	Got	
✓	2	2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 2 x 4 = 8 2 x 5 = 10 2 x 6 = 12 2 x 7 = 14 2 x 8 = 16 2 x 9 = 18 2 x 10 = 20	2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 2 x 4 = 8 2 x 5 = 10 2 x 6 = 12 2 x 7 = 14 2 x 8 = 16 2 x 9 = 18 2 x 10 = 20	✓

Passed all tests! ✓

Question 2

Correct

Marked out of 1.00

Flag question

A nutritionist is labeling all the best power foods in the market. Every food item arranged in a single line, will have a value beginning from 1 and increasing by 1 for each, until all items have a value associated with them. An item's value is the same as the number of macronutrients it has. For example, food item with value 1 has 1 macronutrient, food item with value 2 has 2 macronutrients, and incrementing in this fashion.

The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients. However, the nutritionist must avoid prescribing a particular sum of macronutrients (an 'unhealthy' number), and this sum is known. The nutritionist chooses food items in the increasing order of their value. Compute the highest total of macronutrients that can be prescribed to a patient, without the sum matching the given 'unhealthy' number.

Here's an illustration:

Given 4 food items (hence value: 1,2,3 and 4), and the unhealthy sum being 6 macronutrients, on choosing items 1, 2, 3 -> the sum is 6, which matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is from among:

- $2 + 3 + 4 = 9$
- $1 + 3 + 4 = 8$
- $1 + 2 + 4 = 7$

Since $2 + 3 + 4 = 9$, allows for maximum number of macronutrients, 9 is the right answer.

Complete the code in the editor below. It must return an integer that represents the maximum total of macronutrients, modulo 1000000007 ($10^9 + 7$).

It has the following:

n : an integer that denotes the number of food items

k : an integer that denotes the unhealthy number

Constraints

- $1 \leq n \leq 2 \times 10^9$
- $1 \leq k \leq 4 \times 10^{15}$

Input Format For Custom Testing

The first line contains an integer, n , that denotes the number of food items.

The second line contains an integer, k , that denotes the unhealthy number.

Sample Input 0

2
2

Sample Output 0

3

Explanation 0

The following sequence of $n = 2$ food items:

1. Item 1 has 1 macronutrients.
2. $1 + 2 = 3$; observe that this is the max total, and having avoided having exactly $k = 2$ macronutrients.

Sample Input 1

2
1

Sample Output 1

2

Explanation 1

1. Cannot use item 1 because $k = 1$ and $sum = k$ has to be avoided at any time.
2. Hence, max total is achieved by $sum = 0 + 2 = 2$.

Sample Case 2

Sample Input For Custom Testing

Sample Input 2

3

3

Sample Output 2

5

Explanation 2

2 + 3 = 5, is the best case for maximum nutrients.

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2
3 int main()
4
5 {
6
7     long n,k, sum=0;
8
9     scanf("%ld%ld",&n,&k);
10
11     for(int i=0;i<=n;i++){
12         sum=sum+i;
13
14         if(sum==k){
15             sum-=1;
16         }
17     }
18
19     printf("%ld", sum%1000000007);
20
21 }
22
23
24
25 }
```

	Input	Expected	Got	
✓	2 2	3	3	✓
✓	2 1	2	2	✓
✓	3	5	5	✓

Question 3

Correct

Marked out of
1.00[Flag
question](#)

Determine all positive integer values that evenly divide into a number, its factors. Return the p^{th} element of your list, sorted ascending. If there is no p^{th} element, return 0.

For example, given the number $n = 20$, its factors are $\{1, 2, 4, 5, 10, 20\}$. Using **1-based indexing** if $p = 3$, return 4. If $p > 6$, return 0.

Complete the code in the editor below. The function should return a long integer value of the p^{th} integer factor of n .

It has the following:

n : an integer

p : an integer

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 Input 0

10

3

Sample Output 0

5

Explanation 0

Factoring $n = 10$ we get $\{1, 2, 5, 10\}$. We then return the $p = 3^{\text{rd}}$ factor as our answer.

Sample Input 1

10

5

Sample Output 1

0

Explanation 1

Factoring $n = 10$ we get $\{1, 2, 5, 10\}$. There are only 4 factors and $p = 5$. We return 0 as our answer.

Sample Input 2

1

1

Sample Output 2

1

Explanation 2

Factoring $n = 1$ we get $\{1\}$. We then return the $p = 1^{\text{st}}$ factor as our answer.

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 #include <math.h>
4
5 long long pthFactor (long long n, long long p) {
6
7     long long factors [100000]; // To store factors
8
9     long long count = 0;
10
11     // Find factors of n
12
13     for (long long i = 1; i <= sqrt(n); i++) {
14
15         if (n % i == 0) {
16
17             factors [count++] = i; // Store the factor
18
19             if (i != n / i) {
20
21                 factors [count++] = n / i; // Store the correspondin
22
23             }
24
25         }
26
27     }
28
29     // Sort the factors (simplified because small dataset)
30
31     for (long long i = 0; i < count - 1; i++) {
```

```

32
33 v for (long long j = i + 1; j < count; j++) {
34
35 v if (factors[i] > factors[j]) {
36
37     long long temp = factors[i];
38
39     factors[i] = factors[j];
40
41     factors[j] = temp;
42
43 }
44
45 }
46
47 }
48 // Check if p-th factor exists
49
50 v if (p <= count) {
51
52     return factors[p - 1];

```

	Input	Expected	Got	
✓	10 3	5	5	✓
✓	10 5	0	0	✓
✓	1 1	1	1	✓

Passed all tests! ✓

Finish review