S.Ram Kumar
240901084
EEE -B

WEEK 6

Q1) 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 X i*** (where ***1 ≤ i ≤ 10***) should be printed on a new line in the form: n x i = result.

Input Format

A single integer, ***n***.

Constraints

2 ≤ n ≤ 20

Output Format

Print ***10*** lines of output; each line ***i*** (where ***1 ≤ i ≤ 10***) contains the ***result*** of ***n X i*** in the form:

n x i = result.

Sample Input

2

Sample Output

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

```c
#include <stdio.h>
int main() {
    int n, result, i=1;
    scanf("%d", &n);
    while (i<=10){
        result=n*i;
        printf("%d x %d = %d", n, i, result);
        printf("\n");
        i++;
    }
return 0;
}
```

	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! ✓

Q2) A nutritionist is labelling 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

· $1 \le n \le 2 \times 10^9$

· $1 \le k \le 4 \times 10^{15}$

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, i*s the best case for maximum nutrients.

```c
#include <stdio.h>
int main() {
    long n, k, sum=0;
    scanf("%ld %ld", &n, &k);
    for (int i=1; i<=n; i++) {
        sum+=i;
        if (sum==k) {
            sum-=1;
        }
    }
    printf("%ld",sum%1000000007);
    return 0;
}
```

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

Passed all tests! ✓

Q3) 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

· $1 \le n \le 10^{15}$

· $1 \le p \le 10^{9}$

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 Input 0

10

3

Sample Output 0

5

Explanation 0

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

```c
#include <stdio.h>
int main() {
    int a[100], n, count=0, t, p;
    scanf("%d", &n);
    scanf("%d", &p);
    for (int i=1; i<=n; i++) {
        if (n%i==0) {
            a[count]=i;
            count++;
    }
    }
    for (int i=0; i<(count-1); i++) {
        for (int j=(i+1); j<count; j++) {
            if (a[i]>a[j]) {
                t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
        }
    }
    if (p>0 && p<=count) {
        printf("%d", a[p-1]);
    }
    else {
        printf("%d", 0);
    }
    return 0;
}
```

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

Passed all tests! ✓