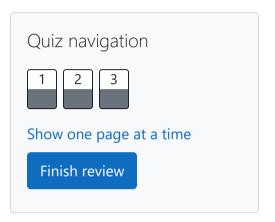
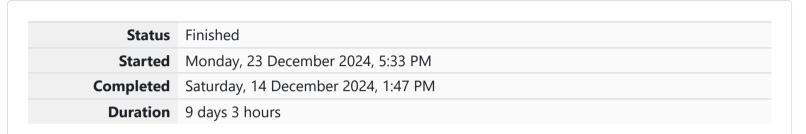
GE23131-Programming Using C-2024





Question **1**

Correct

Marked out of 3.00

Flag question

A set of N numbers (separated by one space) is passed as input to the program. The program must identify the count of numbers where the number is odd number.

Input Format:

The first line will contain the N numbers separated by one space.

Boundary Conditions:

The value of the numbers can be from -99999999 to 99999999

Output Format:

The count of numbers where the numbers are odd numbers.

Input:

5 10 15 20 25 30 35 40 45 50

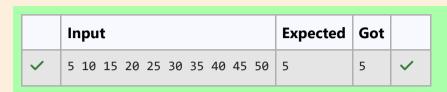
Output:

5

Explanation:

The numbers meeting the criteria are 5, 15, 25, 35, 45.

Answer: (penalty regime: 0 %)



Passed all tests! <

Question **2**

Correct

Marked out of 5.00

Flag question

Given a number N, return true if and only if it is a *confusing number*, which satisfies the following condition:

We can rotate digits by 180 degrees to form new digits. When 0, 1, 6, 8, 9 are rotated 180 degrees, they become 0, 1, 9, 8, 6 respectively. When 2, 3, 4, 5 and 7 are rotated 180 degrees, they become invalid. A *confusing number* is a number that when rotated 180 degrees becomes a **different** number with each digit valid.

Example 1:

6 -> 9

Input: 6

Output: true

Explanation:

We get 9 after rotating 6, 9 is a valid number and 9!=6.

Input: 89

Output: true

Explanation:

We get 68 after rotating 89, 86 is a valid number and 86!=89.

Example 3:

11 -> 11

Input: 11

Output: false

Explanation:

We get 11 after rotating 11, 11 is a valid number but the value remains the same, thus 11 is not a confusing number.

Note:

- 1. $0 <= N <= 10^9$
- 2. After the rotation we can ignore leading zeros, for example if after rotation we have 0008 then this number is considered as just 8.

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
2 v int main(){
    int n,x,y=1;
    scanf("%d",&n);
    while (n!=0&&y==1){
        x=n%10; n=n/10;
        if (x==2||x==3||x==4||x==7){
        y++;
    }
```

	Input	Expected	Got	
~	6	true	true	~
~	89	true	true	~
~	25	false	false	~

Passed all tests! <

Question $\bf 3$

Correct

Marked out of 7.00

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

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

$$\cdot \qquad 1 \le n \le 2 \times 10^9$$

$$1 \le k \le 4 \times 10^{15}$$

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

Sample Output 1						
2						
Explanation 1						
1. Cannot use item 1 because $k = 1$ and $sum \equiv k$ 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						

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
 2 v int main(){
 3
        long long int n,t,i,nut=0;
        scanf("%11d %11d",&n,&t);
       for (i=1;i<=n;i++){
 5 🔻
            nut = nut + i;
           if (nut == t){
 7 ▼
                nut = nut -1;
 8
 9
10
11
        printf("%11d",nut%1000000007);
12 }
```

	Input	Expected	Got	
~	2 2	3	3	~
~	2	2	2	~
~	3	5	5	~

Finish review