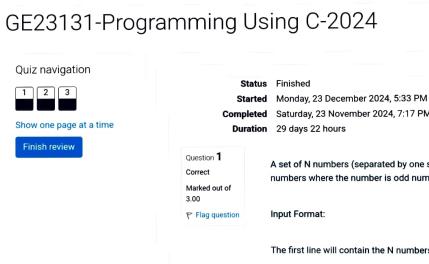
GE23131-Programming Using C-2024

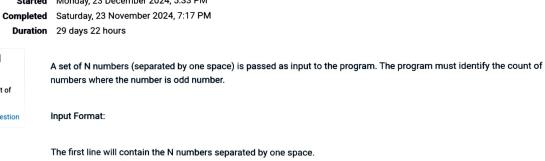
Boundary Conditions:

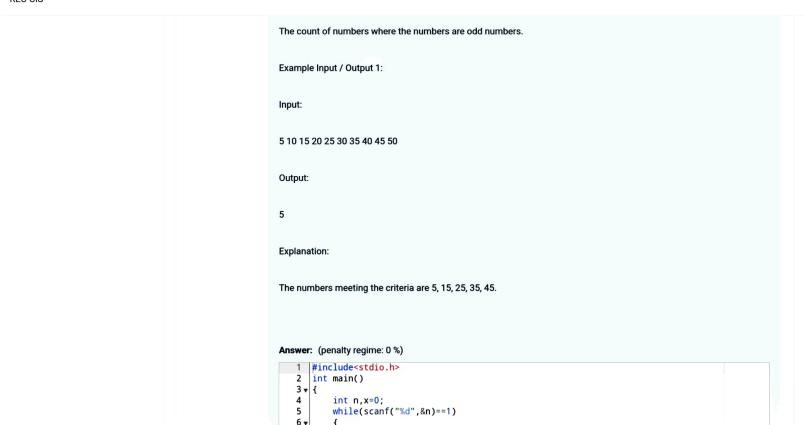
The value of the numbers can be from -99999999 to 99999999

3 <= N <= 50

Output Format:







```
Answer: (penalty regime: 0 %)
     #include<stdio.h>
      int main()
          int n,x=0;
          while(scanf("%d",&n)==1)
              if(n%2!=0)
                  X++;
  10
  11
  12
          printf("%d",x);
  13
          return 0;
  14
  15
```

	Input	Expected	Got	
~	5 10 15 20 25 30 35 40 45 50	5	5	~

6->9 Input: 6 Output: true 89 -> 68 Input: 89 11 -> 11

Question 2 Correct Marked out of

Flag question

5.00

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:

Explanation:

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

Example 2:

Output: true

Explanation:

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

Example 3:

Example 2:

89 -> 68

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

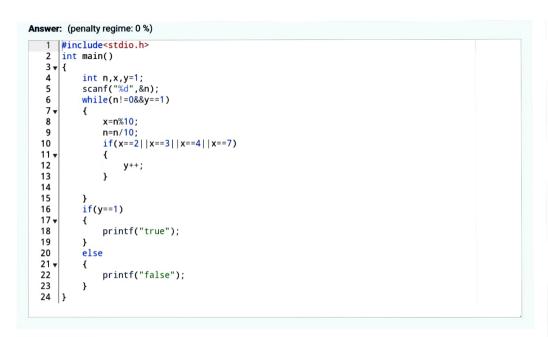
Output: false

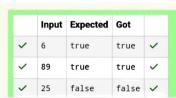
Explanation:

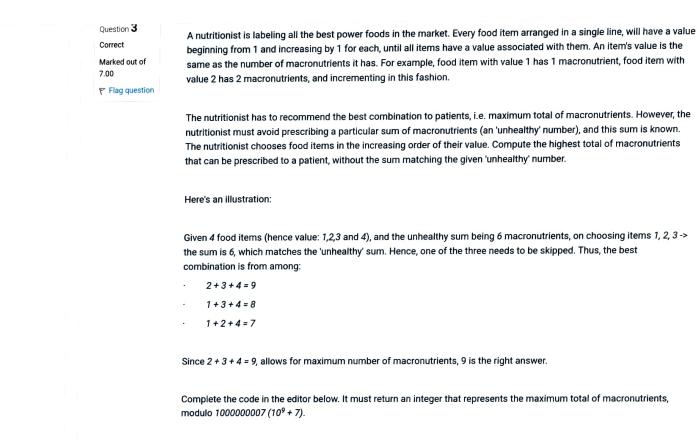
Note:

- 1. 0 <= N <= 10⁹
- 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.

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







It has the following:

n: an integer that denotes the number of food items

k: an integer that denotes the unhealthy number

Constraints

- · 1≤n≤2×109
- $1 \le k \le 4 \times 10^{15}$

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
- .

Sample Output 0

REC-CIS	
	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 $sum \equiv k$ has to be avoided at any time.
	2. Hence, max total is achieved by $sum = 0 + 2 = 2$.

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

Sample Case 2

Sample Input For Custom Testing

Sample Input 2

Sample Output 2

Explanation 2

3

5

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

```
ruionen (penant) regime. o .o/
       |#include<stdio.h>
       int main()
    3 ₹ {
            long long int n,t,i,nut=0;
scanf("%lld %lld",&n,&t);
            for(i=1;i<=n;i++)
                nut=nut+i;
    9
                if(nut==t)
  10 ₩
  11
                     nut=nut-1;
  12
  13
  14
  15
            printf("%lld",nut%1000000007);
            return 0;
  16
  17 }
```

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

