

REC-CIS

GE23131-Programming Using C-2024

Quiz navigation

[Show one page at a time](#)[Finish review](#)

Status	Finished
Started	Monday, 23 December 2024, 5:33 PM
Completed	Friday, 6 December 2024, 9:24 PM
Duration	16 days 20 hours

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 number is odd number.

Input Format:

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

Boundary Conditions:

$3 \leq N \leq 50$

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

Output Format:

The count of numbers where the numbers are odd numbers.

Example Input / Output 1:

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 %)

```
1 #include<stdio.h>
2
3 int main()
4 {
5     int n,x=0;
6     while(scanf("%d",&n)==1)
7     {
8         if(n%2!=0)
9         {
10             x++;
11         }
12     }
13     printf("%d", x);
14 }
```

REC-CIS

17

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

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 conditions:

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 consists of only valid digits and is 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.

Example 2:

89 -> 68

Input: 89

Output: true

Explanation:

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

- 0 <= N <= 10⁹
- After the rotation we can ignore leading zeros, for example if after rotation we have 0001, it is considered as 1.

Answer: (penalty regime: 0 %)

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

REC-CIS

```

13     }
14     }
15     if(y==1)
16     {
17         printf("true");
18     }
19
20     else{
21         printf("false");
22     }
23 }

```

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

Passed all tests!

Question 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 and increasing by 1 for each, until all items have a value associated with them. An item's value has. For example, food item with value 1 has 1 macronutrient, food item with value 2 has 2 mac

The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients (an 'unhealthy' number), and this sum is the increasing order of their value. Compute the highest total of macronutrients that can be recommended 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 matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is:

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

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 number of macronutrients that can be recommended (10⁹ + 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.

REC-CIS

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 $sum \equiv k$ 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

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

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2
3 int main()
4 {
5     long long int n,t,i,nut=0;
6     scanf("%lld %lld", &n, &t);
7     for (i=1;i<=n;i++)
```

REC-CIS

```
10         if(nut==t)
11         {
12             nut=nut-1;
13         }
14     }
15     printf("%lld", nut%1000000007);
16 }
17
```

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

Passed all tests!