SARADHA R 2024-CSE S2

Week-04-Decision Making and Looping - while, do...while and for

Week-04-02-Practice Session-Coding

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:

3 <= N <= 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

Source code

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>

int main(){
    int n,count;
    char ch;

    while(scanf("%d",&n)==1){
        if(n%2!=0){
            count++;
        }
        ch=getchar();
        if(ch=='\n'){
            break;
        }
    }
    printf("%d\n",count);
    return 0;
}
```

Result

	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 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.

Example 2:

89 -> 68

Input: 89

Output: true

Explanation:

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

Source code

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>

int rotated_digit(int digit){
    if(digit==0 || digit==1 || digit==8){
        return digit;
    }
    else if(digit==6){
        return 9;
    }
    else if(digit==9){
        return 6;
    }
    else{
        return -1;
    }
}

int conf(int n){
    int original=n;
    int rotated=0;

    while(n>0){
        int digit=n%10;
        rotated=rotated_digit(digit);
        if(rotated==-1){
            return 0;
        }
        rotated=rotated*10+rotated;
        n/=10;
    }
    return rotated!=original;
}

int main(){
    int n;

    scanf("%d",&n);

    if(conf(n)){
        printf("true");
    }
    else{
        printf("false");
    }
    return 0;
}
```

Result

	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 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

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

Source code

Answer: (penalty regime: 0 %)

```
#include<stdio.h>
#define mod 1000000007

int main(){
    int n,k;
    long long sum=0;

    scanf("%d\n%d",&n,&k);

    for(int i=1;i<=n;i++){
        sum+=i;
        sum=(sum+mod%mod)%mod;
        if(sum==k){
            sum-=1;
        }
    }
    printf("%lld",sum);
    return 0;
}
```

Result

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

Passed all tests! ✓