Assignment #3: 语法练习

Updated 1600 GMT+8 Sep 25, 2023

2023 fall, Complied by Xinjie Song, Phy

说明：

1）第2周课上讲到了计算机相关的历史，介绍了ASCII表。

2）请把每个题目解题思路（可选），源码Python, 或者C++/C（已经在Codeforces/Openjudge上AC），截图（包含Accepted, 学号），填写到下面作业模版中（推荐使用 typora https://typoraio.cn，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。

3）提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、作业评论有md或者doc。

4）同学完成作业的时候，就是这个模版文件中修改补充好。为便于助教批改作业，请尽量不要删除其他文字。

5）如果不能在截止前提交作业，请写明原因。

编程环境

操作系统：Windows 11 22H2

Python编程环境：PyCharm 2023.2 (Community Edition)

C/C++编程环境：g++ (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0

1. 必做题目

118A. String Task

implementation/strings, 1000, http://codeforces.com/problemset/problem/118/A

Petya started to attend programming lessons. On the first lesson his task was to write a simple program. The program was supposed to do the following: in the given string, consisting if uppercase and lowercase Latin letters, it:

- deletes all the vowels,
- inserts a character "." before each consonant,
- replaces all uppercase consonants with corresponding lowercase ones.

Vowels are letters "A", "O", "Y", "E", "U", "I", and the rest are consonants. The program's input is exactly one string, it should return the output as a single string, resulting after the program's processing the initial string.

Help Petya cope with this easy task.

Input

The first line represents input string of Petya's program. This string only consists of uppercase and lowercase Latin letters and its length is from 1 to 100, inclusive.

Output

Print the resulting string. It is guaranteed that this string is not empty.

Examples

input

```
tour
```

output

```
.t.r
```

input

```
Codeforces
```

output

```
.c.d.f.r.c.s
```

input

```
aBAcAba
```

output

```
.b.c.b
```

【宋昕杰，物理学院，2023年秋】

思路：利用in关键字判断某个字符是否是元音字母，然后按要求更改即可

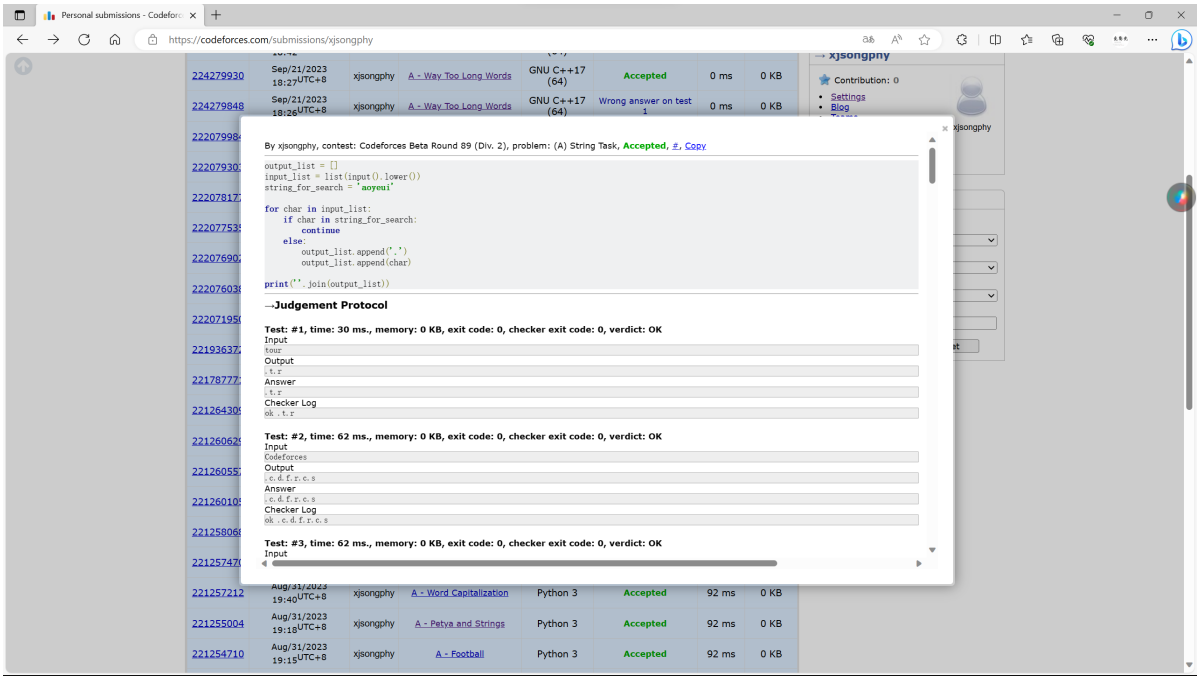
代码

```
output_list = []
input_list = list(input().lower())
string_for_search = 'aoyeui'

for char in input_list:
    if char in string_for_search:
        continue
    else:
        output_list.append('.')
        output_list.append(char)

print(''.join(output_list))
```

代码运行截图



263A. Beautiful Matrix

implementation, 800, http://codeforces.com/problemset/problem/263/A

You've got a 5 × 5 matrix, consisting of 24 zeroes and a single number one. Let's index the matrix rows by numbers from 1 to 5 from top to bottom, let's index the matrix columns by numbers from 1 to 5 from left to right. In one move, you are allowed to apply one of the two following transformations to the matrix:

1. Swap two neighboring matrix rows, that is, rows with indexes i and $i + 1$ for some integer i ($1 \le i < 5$).
2. Swap two neighboring matrix columns, that is, columns with indexes j and $j + 1$ for some integer j ($1 \le j < 5$).

You think that a matrix looks *beautiful*, if the single number one of the matrix is located in its middle (in the cell that is on the intersection of the third row and the third column). Count the minimum number of moves needed to make the matrix beautiful.

Input

The input consists of five lines, each line contains five integers: the *j*-th integer in the *i*-th line of the input represents the element of the matrix that is located on the intersection of the *i*-th row and the *j*-th column. It is guaranteed that the matrix consists of 24 zeroes and a single number one.

Output

Print a single integer — the minimum number of moves needed to make the matrix beautiful.

Examples

input

```
0 0 0 0 0
0 0 0 0 1
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

output

```
3
```

input

```
0 0 0 0 0
0 0 0 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 0 0
```

output

```
1
```

【宋昕杰，物理学院，2023年秋】

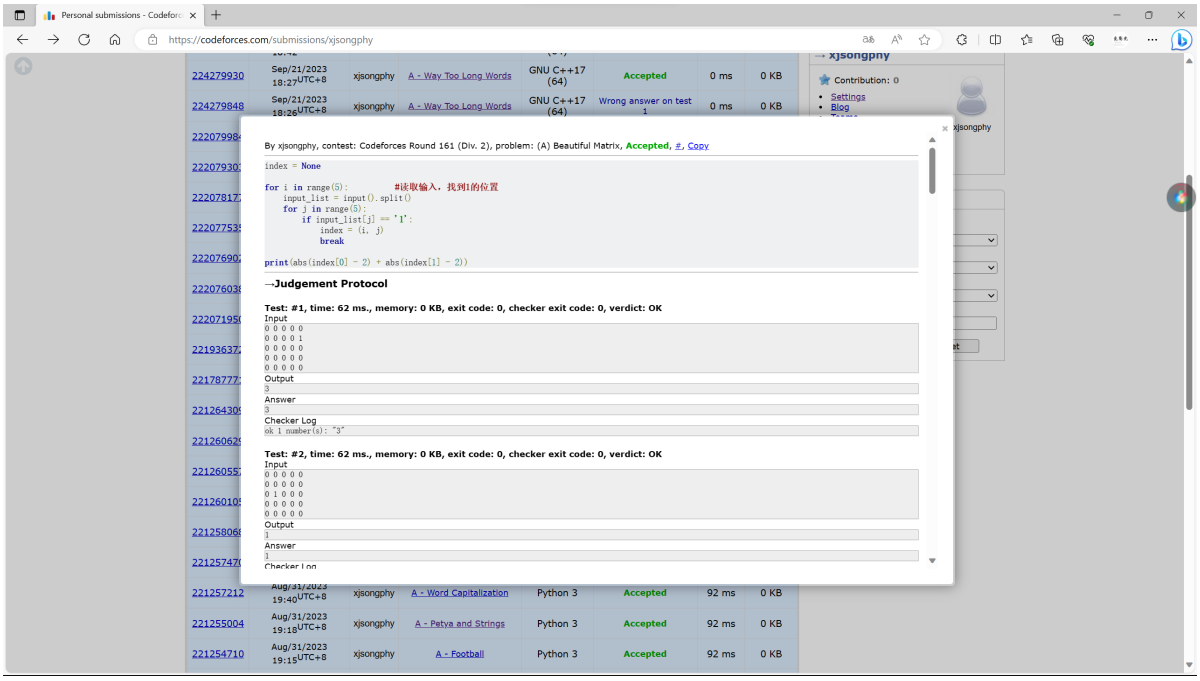
思路：不记录矩阵，只记录'1'的位置，并根据位置计算步数

代码

```
index = None
for i in range(5):            #读取输入，找到1的位置
    input_list = input().split()
    for j in range(5):
        if input_list[j] == '1':
            index = (i, j)
            break

print(abs(index[0] - 2) + abs(index[1] - 2))
```

代码运行截图



281A. Word Capitalization

implementation/strings, 800, http://codeforces.com/problemset/problem/281/A

Capitalization is writing a word with its first letter as a capital letter. Your task is to capitalize the given word.

Note, that during capitalization all the letters except the first one remains unchanged.

Input

A single line contains a non-empty word. This word consists of lowercase and uppercase English letters. The length of the word will not exceed 10^3^.

Output

Output the given word after capitalization.

Examples

input

```
ApPLe
```

output

```
ApPLe
```

input

```
konjac
```

output

```
Konjac
```

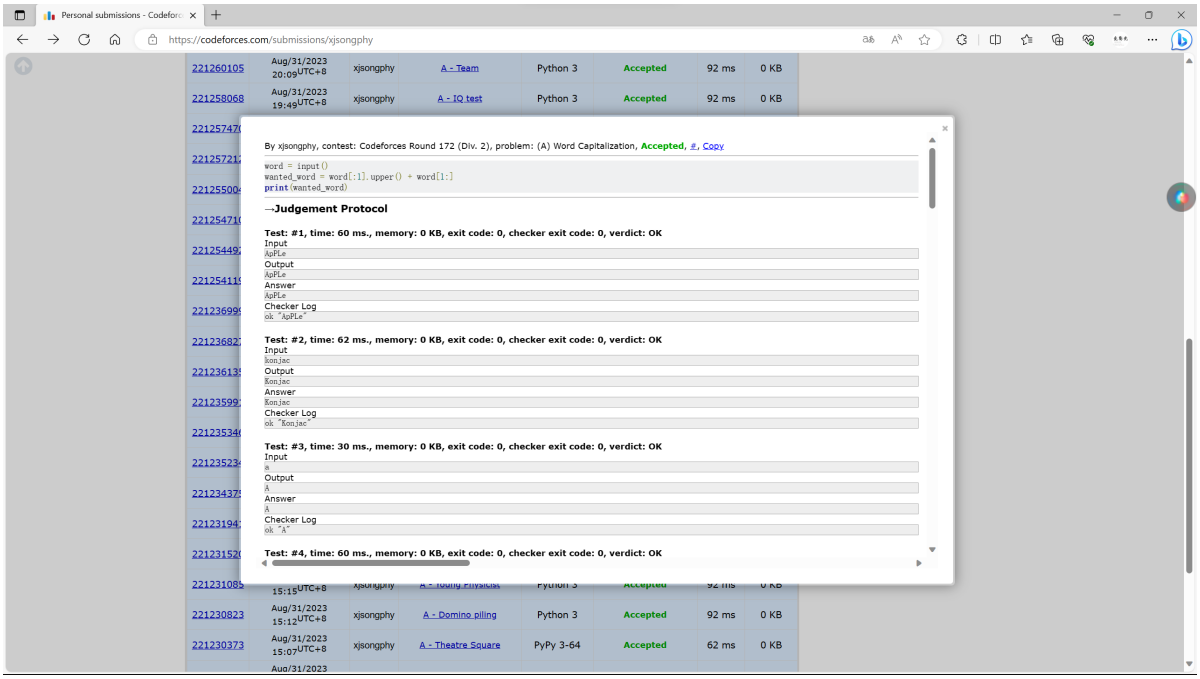
【宋昕杰，物理学院，2023年秋】

思路：利用切片和upper()方法将首字母大写

代码

```
word = input()
wanted_word = word[:1].upper() + word[1:]
print(wanted_word)
```

代码运行截图



282A. Bit++

implementation, 800, http://codeforces.com/problemset/problem/282/A

The classic programming language of Bitland is Bit++. This language is so peculiar and complicated.

The language is that peculiar as it has exactly one variable, called *x*. Also, there are two operations:

- Operation ++ increases the value of variable *x* by 1.
- Operation -- decreases the value of variable *x* by 1.

A statement in language Bit++ is a sequence, consisting of exactly one operation and one variable *x*. The statement is written without spaces, that is, it can only contain characters "+", "-", "X". Executing a statement means applying the operation it contains.

A programme in Bit++ is a sequence of statements, each of them needs to be executed. Executing a programme means executing all the statements it contains.

You're given a programme in language Bit++. The initial value of *x* is 0. Execute the programme and find its final value (the value of the variable when this programme is executed).

Input

The first line contains a single integer n ($1 \le n \le 150$) — the number of statements in the programme.

Next n lines contain a statement each. Each statement contains exactly one operation (++ or --) and exactly one variable *x* (denoted as letter «X»). Thus, there are no empty statements. The operation and the variable can be written in any order.

Output

Print a single integer — the final value of *x*.

Examples

input

```
1
++X
```

output

```
1
```

input

```
2
X++
--X
```

output

```
0
```

【宋昕杰，物理学院，2023年秋】

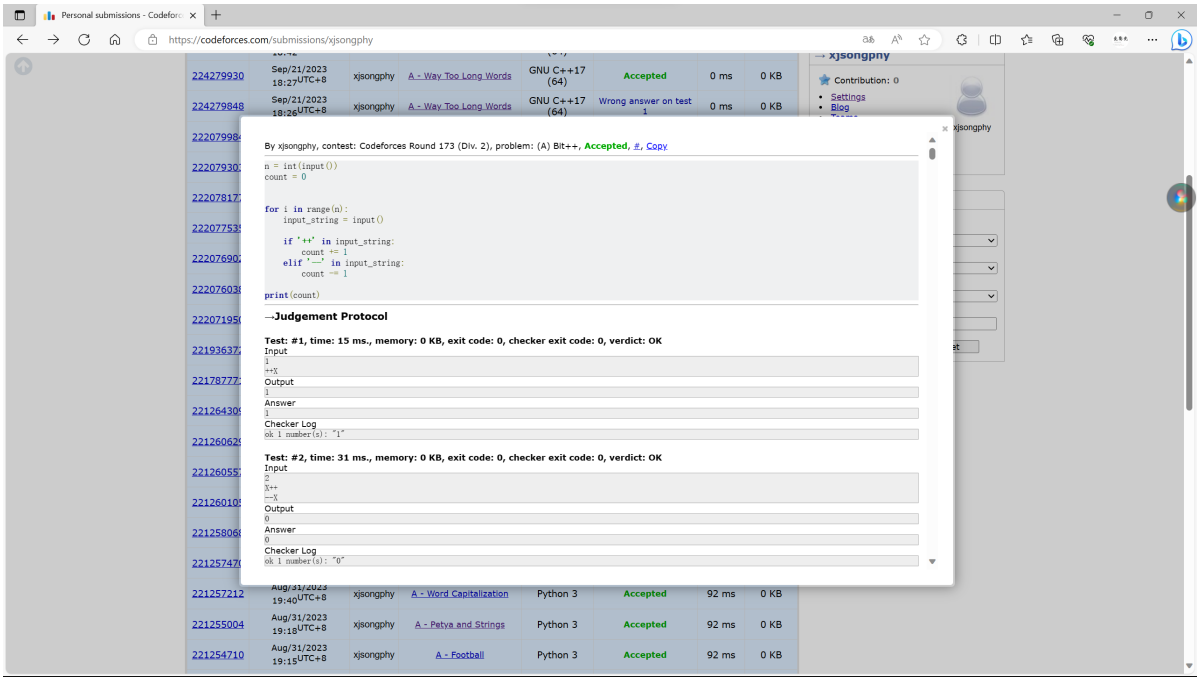
思路：利用in关键字判断'++'或'--'是否在输入中，并记录

代码

```
n = int(input())
count = 0
for i in range(n):
    input_string = input()

    if '++' in input_string:
        count += 1
    elif '--' in input_string:
        count -= 1
print(count)
```

代码运行截图



339A. Helpful Maths

greedy/implementation/sortings/strings, 800, http://codeforces.com/problemset/problem/339/A

Xenia the beginner mathematician is a third year student at elementary school. She is now learning the addition operation.

The teacher has written down the sum of multiple numbers. Pupils should calculate the sum. To make the calculation easier, the sum only contains numbers 1, 2 and 3. Still, that isn't enough for Xenia. She is only beginning to count, so she can calculate a sum only if the summands follow in non-decreasing order. For example, she can't calculate sum 1+3+2+1 but she can calculate sums 1+1+2 and 3+3.

You've got the sum that was written on the board. Rearrange the summans and print the sum in such a way that Xenia can calculate the sum.

Input

The first line contains a non-empty string *s* — the sum Xenia needs to count. String *s* contains no spaces. It only contains digits and characters "+". Besides, string *s* is a correct sum of numbers 1, 2 and 3. String *s* is at most 100 characters long.

Output

Print the new sum that Xenia can count.

Examples

input

```
3+2+1
```

output

```
1+2+3
```

input

```
1+1+3+1+3
```

output

```
1+1+1+3+3
```

input

```
2
```

output

```
2
```

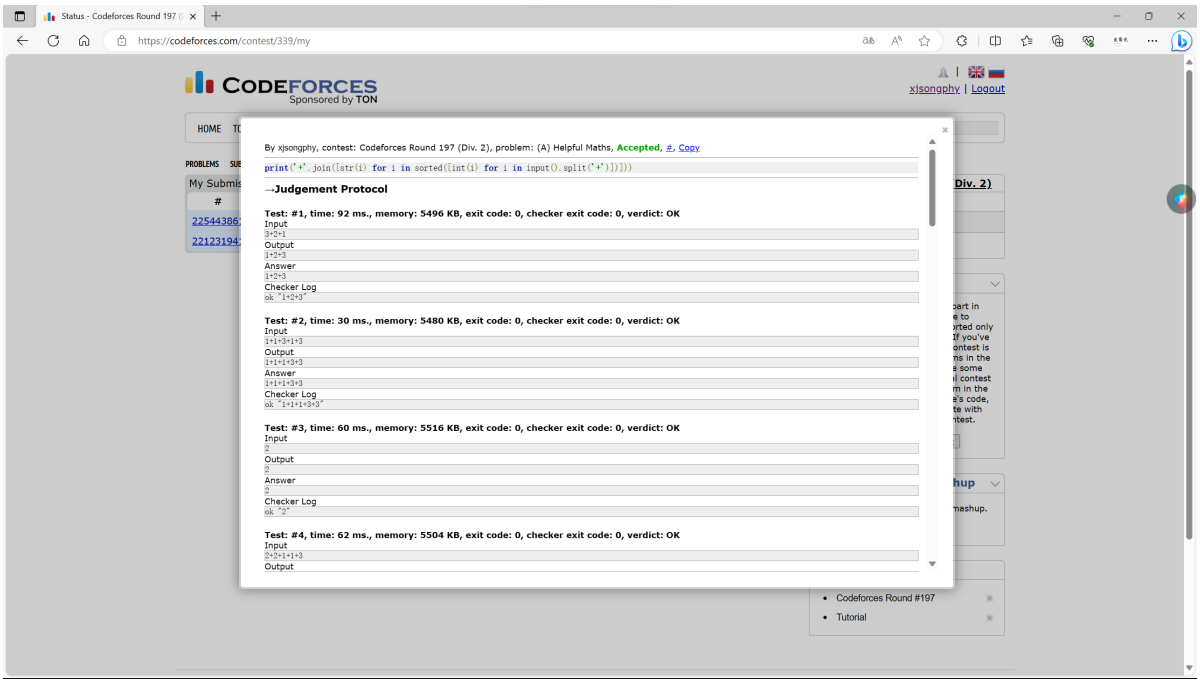
【宋昕杰，物理学院，2023年秋】

思路：利用split('+')将输入存储为列表并排序，再用'+'连接即可

代码

```
print('+'.join([str(i) for i in sorted([int(i) for i in input().split('+')])]))
```

代码运行截图



2. 选做题目

OJ01017: 装箱问题

greedy, http://cs101.openjudge.cn/practice/01017

一个工厂制造的产品形状都是长方体，它们的高度都是h，长和宽都相等，一共有六个型号，他们的长宽分别为1*1, 2*2, 3*3, 4*4, 5*5, 6*6。这些产品通常使用一个 6*6*h 的长方体包裹包装然后邮寄给客户。因为邮费很贵，所以工厂要想方设法的减小每个订单运送时的包裹数量。他们很需要有一个好的程序帮他们解决这个问题从而节省费用。现在这个程序由你来设计。

输入：输入文件包括几行，每一行代表一个订单。每个订单里的一行包括六个整数，中间用空格隔开，分别为*1*1至6*6*这六种产品的数量。输入文件将以6个0组成的一行结尾。

输出：除了输入的最后一行6个0以外，输入文件里每一行对应着输出文件的一行，每一行输出一个整数代表对应的订单所需的最小包裹数。

解题思路：4*4, 5*5, 6*6这三种的处理方式较简单，就是每一个箱子至多只能有其中1个，根据他们的数量添加箱子，再用2*2和1*1填补。1*1, 2*2, 3*3这些就需要额外分情况讨论，若有剩余的3*3,每4个3*3可以填满一个箱子，剩下的3*3用2*2和1*1填补装箱。剩余的2*2，每9个可以填满一个箱子，剩下的与1*1一起装箱。最后每36个1*1可以填满一个箱子，剩下的为一箱子。

样例输入

```
0 0 4 0 0 1
7 5 1 0 0 0
0 0 0 0 0 0
```

样例输出

```
2
1
```

来源：Central Europe 1996

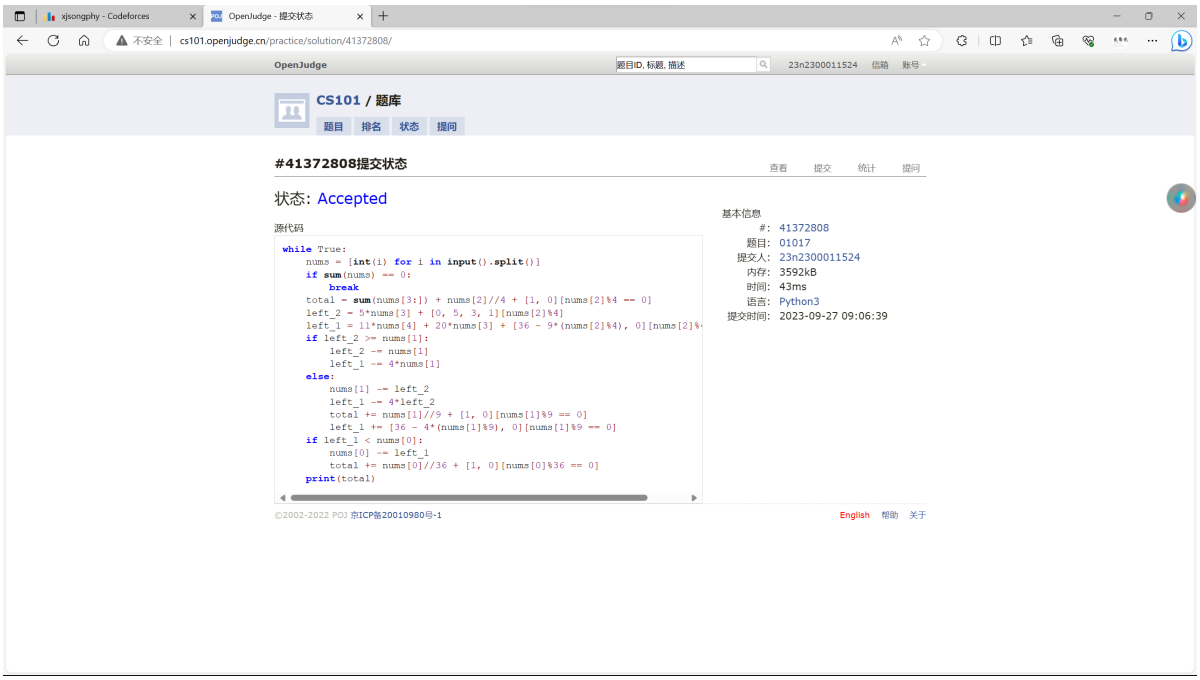
【宋昕杰，物理学院，2023年秋】

思路：上述贪心思路

代码

```
while True:
    nums = [int(i) for i in input().split()]
    if sum(nums) == 0:
        break
    total = sum(nums[3:]) + nums[2]//4 + [1, 0][nums[2]%4 == 0]
    left_2 = 5*nums[3] + [0, 5, 3, 1][nums[2]%4]
    left_1 = 11*nums[4] + 20*nums[3] + [36 - 9*(nums[2]%4), 0][nums[2]%4 == 0]
    if left_2 >= nums[1]:
        left_2 -= nums[1]
        left_1 -= 4*nums[1]
    else:
        nums[1] -= left_2
        left_1 -= 4*left_2
        total += nums[1]//9 + [1, 0][nums[1]%9 == 0]
        left_1 += [36 - 4*(nums[1]%9), 0][nums[1]%9 == 0]
    if left_1 < nums[0]:
        nums[0] -= left_1
        total += nums[0]//36 + [1, 0][nums[0]%36 == 0]
    print(total)
```

代码运行截图



3. 学习总结和收获

所有题目均为提前完成，整理作业过程中发现CF339A当时的解法有多条语句没有意义，遂合并，最终成功将整个程序缩为一条语句；合理利用列表可以减少代码量，但易读性不佳。

每日尽可能完成新增题目，但不排除时间紧张，没有完成。

截至2023年9月27日09：00，OJ完成题目47道，CF完成题目25道。