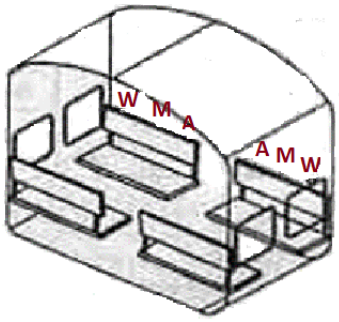
Черных Владимир Артемович

Балл: 130

Задача №1

Условие

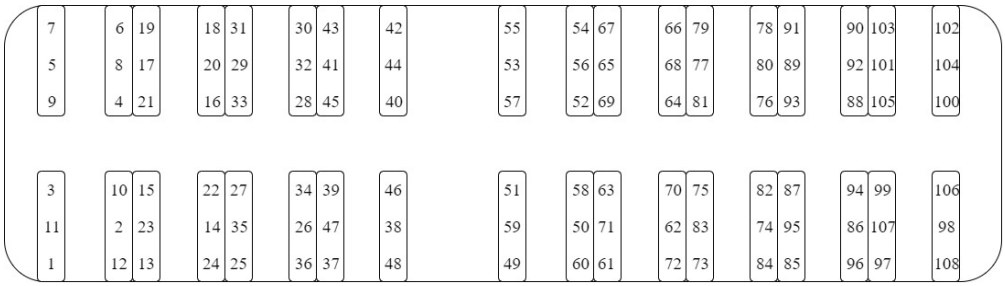
Наташа и **Олег** очень любят путешествовать. В основном они путешествуют по железной дороге. Однажды они ехали в вагоне, где были только сидячие места, и заинтересовались расположением сидений в своем купе. Купе выглядело примерно так



Тип места обозначается следующим образом:

- Сиденье у окна: **W**
- Среднее место: **M**
- Место у прохода: **A**

Нумерация мест в вагоне приведена на следующем рисунке



Ребятам стало интересно узнать номер и тип сиденья, расположенного рядом с ними. В случае, когда сиденье расположено в середине, необходимо указать оба соседних номера.

Напишите программу, которая поможет ребятам решить эту задачу.

Входные данные

На вход подается одно целое число **N (**1 <= **N** <= 108**),** обозначающее номер места.

Выходные данные

Целое число, и большая латинская буква **W, M** или **A**, разделенные одним пробелом, которые будут соответствовать номеру и типу соседнего сиденья. Если таких сидений два, то данные по второму сидению выводится в той же строке через пробел (номера сидений в этом случае выводятся в порядке возрастания).

Примеры

Входные данные	Выходные данные
19	17 M
29	31 W 33 A

Исходный код

```
map_ = {1: [[11, 'M']],
        11:[[1, 'W'], [3, 'A']],
        3: [[11, 'M']],

        9: [[5, 'M']],
        5:[[7, 'W'], [9, 'A']],
        7: [[5, 'M']],

        6: [[8, 'M']],
        8:[[6, 'W'], [4, 'A']],
        4: [[8, 'M']],

        10: [[2, 'M']],
        2:[[12, 'W'], [10, 'A']],
        12: [[2, 'M']]
        }


n = int(input())
x = (n - 1) // 12
n %= 12
n = n if n > 0 else 12
ans = map_[n][:]
for i in range(len(ans)):
    ans[i][0] += 12 * x
ans.sort(key=lambda q: q[0])
for elem in ans:
    print(*elem, end=' ')
```

Задача №2

Условие

В кассе, продающей билеты по цене 500 р., нет возможности расплатиться картой, а в начале работы у кассира для сдачи есть только ***m*** купюр по 500 р. Перед открытием у кассы собрались посетители, у каждого человека в очереди есть только одна купюра. У части – 500 р., у части – 1000 р. Сколькими способами можно выстроить посетителей в очередь так, чтобы к моменту обслуживания посетителя с купюрой в 1000 рублей у кассира всегда была сдача?

Напишите программу, которая решит эту задачу.

На вход программы подаются три целых неотрицательных числа:

n - количество человек с купюрой 500 р,

k - количество человек с купюрой 1000 р

m - количество купюр по 500 р у кассира к началу работы кассы.

Каждое число меньше 10.

Выведите одно целое число – сколькими способами можно выстроить очередь.

Пример

Входные данные	Выходные данные
1 2 1	4

Комментарий к примеру. Пусть есть посетители **A**, **B**, и **C**. У **A** и **B** по 1000 рублёвой купюре, а у **C** – купюра в 500 рублей. Тогда их можно выстроить так:

ABC – нельзя, т.к. единственная купюра 500р. уйдет на сдачу посетителю **A**, а для посетителя **B** сдачи не будет.

ACB – можно, т.к. единственная купюра 500р. уйдет на сдачу посетителю **A,** затем посетитель **C** заплатит купюрой 500р., и для покупателя **B** будет сдача.

BAC – нельзя, т.к. единственная купюра 500р уйдет на сдачу посетителю **B**, а для посетителя **A** сдачи не будет.

BCA – можно

CAB – можно

CBA – можно

Есть только 4 возможных варианта.

Исходный код

```
from math import factorial

n, k, m = map(int, input().split())
cnt = 0
for mask in range(1 << (n + k)):
    bal = m
    if bin(mask)[2:].count('1') == k:
        for i in range(n + k):
            if (mask >> i) & 1 == 0:
                bal += 1
            else:
                bal -= 1
            if bal < 0:
                break
        if bal >= 0:
            cnt += 1
print(cnt * factorial(n) * factorial(k))
```

Задача №3

Условие

Исполнитель получает на вход натуральное число X (не превышающее 10^6). По этому числу, точнее по его представлению в восьмеричной системе счисления, строится новое число Y по следующим правилам.

В восьмеричном представлении числа X предпоследняя цифра увеличивается на 1 (гарантируется, что в восьмеричном представлении X числа больше 2-х цифр). Например, $695_{10} = 12\mathbf{6}7_8$ -> $12\mathbf{7}7_8 = 703_{10}$.

Если предпоследняя цифра 7, тогда предпоследняя цифра становится 0, а последняя изменяется по следующему принципу: четная увеличивается н 1, а нечетная уменьшается на 1. Например, последняя цифра нечетная $697_{10} = 12\mathbf{7}\boldsymbol{1}_8$ -> $12\mathbf{0}\boldsymbol{0}_8 = 640_{10}$, последняя цифра четная $698_{10} = 12\mathbf{7}\boldsymbol{2}_8$ -> $12\mathbf{0}\boldsymbol{3}_8 = 643_{10}$.

Введем понятие расстояния

Oh = | Исходное_число – Полученное_число |

Напишите программу, которая будет считать наибольшее расстояние Oh для чисел из заданного интервала [A,B] и наибольшее исходное число, для которого оно было вычислено.

На вход программы подаётся два целых числа A и B ($10 \le A \le B \le 1\,000\,000$), записанных через пробел.

Программа должна вывести два числа наибольшее расстояние Oh и через пробел исходное число, для которого оно было посчитано.

Входные данные	Вывод	Примечание

694 698	57 697	$\lvert 694 - 702 \rvert = 8$ $(12\mathbf{6}6_8 - 12\mathbf{7}6_8)$ $\lvert 695 - 703 \rvert = 8$ $(12\mathbf{6}7_8 - 12\mathbf{7}7_8)$ $\lvert 696 - 541 \rvert = 55$ $(12\mathbf{7}0_8 - 12\mathbf{0}1_8)$ $\lvert 697 - 640 \rvert = 57$ $(12\mathbf{7}1_8 - 12\mathbf{0}0_8)$ $\lvert 698 - 643 \rvert = 55$ $(12\mathbf{7}2_8 - 12\mathbf{0}3_8)$
693 695	8 695	$\lvert 693 - 701 \rvert = 8$ $(12\mathbf{6}5_8 - 12\mathbf{7}5_8)$ $\lvert 694 - 702 \rvert = 8$ $(12\mathbf{6}6_8 - 12\mathbf{7}6_8)$ $\lvert 695 - 703 \rvert = 8$ $(12\mathbf{6}7_8 - 12\mathbf{7}7_8)$

Исходный код

```
def int2num(n):
    a = []
    while n:
        a.append(n % 8)
        n //= 8
    return a[::-1]

def num2int(a):
    ans = 0
    for elem in a:
        ans = ans * 8 + elem
    return ans


def gen(n):
    s = int2num(n)
    if s[-2] == 7:
        s[-2] = 0
        s[-1] = s[-1] + 1 - 2 * (s[-1] % 2)
    else:
        s[-2] += 1
    return num2int(s)


def oh(n):
    return abs(n - gen(n))

a, b = map(int, input().split())
q = [(oh(num), num) for num in range(a, b + 1)]
ans = max(q)
print(*ans)
```

Задача №4

Условие

Сообщение, которое передают по каналу связи, состоит из чисел, записанных в десятичной системе счисления. Каждое число состоит из шести знаков, а его двоичная запись оканчивается тремя нулями. При передаче сообщение было засорено посторонними шумами: числами, отличающимися от тех, что были в сообщении. Найдите изначальное количество чисел в сообщении.

Формат ввода

В строке вводится сначала целое число $\boldsymbol{n}$ – количество чисел в сообщении ($\boldsymbol{n \le 1000}$), затем $\boldsymbol{n}$ натуральных чисел, все числа отделены друг от друга одним или несколькими пробелами.

Формат вывода

Вывести одно целое число – количество достоверных сигналов в сообщении.

Пример

Входные данные	Выходные данные
3 100128 4356 234064	2

Исходный код

```
n, *a = map(int, input().split())
print(sum([(len(str(elem)) == 6 and elem & ((1 << 3) - 1) == 0) for elem in a]))
```

Задача №5

Условие

Несколько агентов пересылают кодовые сообщения в Центр. Сообщение каждого агента представляет собой несколько слов, к каждому из которых приписан его идентификатор, состоящий из одной буквы. Сообщения записаны по очереди. Найдите сообщение, содержащее больше всего слов.

Формат ввода

В первой строке вводится сначала целое число ***n*** – количество слов (***n ≤ 1000***), затем в ***n*** следующих строках записано по слову. Слова состоят только из строчных латинских букв.

Формат вывода

Вывести одно целое число – длину сообщения (количество слов в сообщении), содержащего больше всего слов.

Примеры

Входные данные	Выходные данные
4 bcd bcfd bcdfe abc	2
3 abd abdc bvd	1

Исходный код

```
n = int(input())
if n:
    ans = 1
    cnt = 1
    char = input()[-1]
    for _ in range(n - 1):
        s = input()[-1]
        if s != char:
            ans = max(ans, cnt)
            cnt = 1
            cnt = 0
        char = s
        cnt += 1
    ans = max(ans, cnt)
    print(ans)
else:
    print(0)
'''
4
bcd
bcfd
bcdfe
abc

3
```

```
abd
abdc
bvd

'''
```

Задача №6

Условие

Павел ведет дневник, наблюдая за погодой. Каждый день он записывает температуру. Требуется определить, в какой день во время периода наблюдений была наименьшая температура и сколько дней отмечалась температура выше порогового значения

На вход программе в первой строке подаются натуральное число N – количество дней, в течение которых велось наблюдение ($N \leq 20$) и вещественное число T – пороговое значение температуры.

Далее в N строках подается на вход по вещественному числу – t_i: Температура в i-й день.

Вывести два целых числа – в первой строке вывести номер дня, в который отмечалась наименьшая температура, во второй строке вывести, сколько дней отмечалась температура выше порогового значения. Если наименьшая температура наблюдалась в несколько разных дней, вывести наименьший номер дня.

Пример

Ввод	Вывод
3 5 5 6 7	1 2

Исходный код

```
n, t = map(int, input().split())
a = [int(input()) for  _ in range(n)]
print(a.index(min(a)) + 1, sum([elem > t for elem in a]), sep='\n')
```

Задача №7

Условие

Робот может выполнять команды «Поиск дефектов», «Движение», «Подготовка» и «Ремонт». Из-за конструктивных особенностей на робота наложены некоторые ограничения. Два раза подряд можно выполнить только команду «Движение». Команда «Ремонт» может быть выполнена только на следующем шаге после команды «Подготовка».

Напишите программу, которая определит, сколько существует выполнимых последовательностей команд длиной n, если до начала выполнения программы робот выполнил команду «Подготовка».

На вход программе подается натуральное число n ($n \leq 15$) – количество команд.

Вывести целое число – количество выполнимых последовательностей команд длиной n.

Пример

Ввод	Вывод
2	8

Исходный код

```
n = int(input())
dp = [{'repair': 1, 'move': 1, 'prepair': 0, 'find' : 1} for _ in range(n)]
for  i in range(1, n):
    dp[i]['repair'] = dp[i - 1]['prepair']
```

```
    dp[i]['move'] = sum(dp[i - 1].values())
    dp[i]['prepair'] = dp[i - 1]['repair'] + dp[i - 1]['move'] + dp[i - 1]['find']
    dp[i]['find'] = dp[i - 1]['repair'] + dp[i - 1]['move'] + dp[i - 1]['prepair']
print(sum(dp[-1].values()))
```