МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИНФОРМАТИКА. ОТБОРОЧНЫЙ ТУР 10 КЛАСС

Часть 1

Соревнования 1 вариант

В группе по программированию имеется 12 учеников. Учитель хочет отправить своих учеников на соревнования. Соревнования проходят в течении двух дней. В каждом дне участвует команда из 5 человек. Сколькими способами учитель может набрать учеников для выступления так, чтобы состав команды первого дня выступлений не совпадал с составом команды второго дня? (Составы не совпадают, если они отличаются хотя бы одним учеником)

Ответ:

626472

Решение:

$C_{12}^{5}(C_{12}^{5} - 1) = 626472$

Соревнования 2 вариант

В группе по программированию имеется 11 учеников. Учитель хочет отправить своих учеников на соревнования. Соревнования проходят в течении двух дней. В каждом дне участвует команда из 5 человек. Сколькими способами учитель может набрать учеников для выступления так, чтобы состав команды первого дня выступлений не совпадал с составом команды второго дня? (Составы не совпадают, если они отличаются хотя бы одним учеником)

Ответ:

212982

Решение:

$C_{11}^{5}(C_{11}^{5} - 1) = 212982$

Минимальная система счисления 1 вариант

Пусть x и y - целочисленные основания двух систем счисления, не превосходящих 36. Дано уравнение $111_x = 333_y$. Найдите такое минимальное x, при котором уравнение верно для некоторого y.

Ответ:

16

Решение:

Уравнение эквивалентно следующему:

$X^2 + X + 1 = 3Y^2 + 3Y + 3$

В записи правого числа есть цифра 3, поэтому минимальное y – 4.

Далее:

```
for x in range(1, 37):
    flag = False
    for y in range(4, 37):
        if x ** 2 + x + 1 == 3 * y ** 2 + 3 * y + 3:
            flag = True
            print(x)
            break
    if flag:
        break
```

Минимальная система счисления 2 вариант

Пусть *x* и *y* - целочисленные основания двух систем счисления, не превосходящих 36. Дано уравнение $111_x = 337_y$. Найдите такое минимальное *x*, при котором уравнение верно для некоторого *y*.

Ответ:

23

Решение:

Уравнение эквивалентно следующему:

$X^2 + X + 1 = 3Y^2 + 3Y + 7$

В записи правого числа есть цифра 7, поэтому минимальное y – 8.

Далее:

```
for x in range(1, 37):
    flag = False
    for y in range(8, 37):
        if x ** 2 + x + 1 == 3 * y ** 2 + 3 * y + 7:
            flag = True
            print(x)
            break
    if flag:
        break
```

Больше делителей 1 вариант

Найдите трёхзначное натуральное число имеющее больше всего делителей среди всех трёхзначных натуральных чисел. Если таких чисел несколько, в ответ напишите минимальное.

Ответ:

840

Решение:

Используя основную теорему арифметики и её свойство про количество делителей – необходимо иметь как можно больше множителей в каноническом разложении, получаем:

$2^3 * 3 * 5 * 7 = 840$

Такое разложение даёт наибольшее количество делителей, поскольку при увеличении степени какого-либо простого числа в разложении на единицу, количество делителей увеличивается в (n + 2) / (n + 1) раз. Например у числа $8 = 2^3$ – четыре делителя (3 + 1). А у числа 2^4 – пять. Но добавление нового простого числа в разложение увеличивает количество делителей сразу в два раза. Поэтому выгодно добавлять новые простые числа в разложение, чем увеличивать степени у уже имеющихся простых чисел.

Больше делителей 2 вариант

Найдите трёхзначное натуральное число имеющее больше всего делителей среди всех трёхзначных натуральных чисел. Если таких чисел несколько, выберите среди них минимальное. В ответ напишите количество делителей найденного числа.

Ответ:

32

Решение:

Используя основную теорему арифметики и её свойство про количество делителей – необходимо иметь как можно больше множителей в каноническом разложении, получаем:

$2^3 * 3 * 5 * 7 = 840$

Данное число имеет 32 делителя.

Такое разложение даёт наибольшее количество делителей, поскольку при увеличении степени какого-либо простого числа в разложении на единицу, количество делителей увеличивается в (n + 2) / (n + 1) раз. Например у числа $8 = 2^3$ – четыре делителя (3 + 1). А у числа 2^4 – пять. Но добавление нового простого числа в разложение увеличивает количество делителей сразу в два раза. Поэтому выгодно добавлять новые простые числа в разложение, чем увеличивать степени у уже имеющихся простых чисел.

Часть 2

На уроке Информатики

Кузьма придумал формулы преобразования чисел.

На каждом шаге Кузьма может выполнить преобразование числа по одной из двух формул:

- $K_{i+1} = K_i + 3$
- $K_{i+1} = K_i * 3$

Кузьма понимает, что формулы имеют недостаток: исходное число не всегда можно преобразовать в заданное конечное. Тем не менее он хочет проверить пары чисел на возможность их преобразования одно в другое.

Формат ввода

В первой строчке содержатся два числа m, n ($0 \le m < n \le 10^6$)- начальное и конечные числа

Формат вывода

Выведите: YES - если из m можно получить n с помощью формул

Иначе выведите: NO

Пример 1

Ввод	Вывод
7 24	YES

Пример 2

Ввод	Вывод
5 16	NO

Пример решения

```python
def solution(num1, num2):
    if (num2 - num1) % 3 == 0:
        return 'YES'
    elif num2 >= 3 * num1 and num2 % 3 == 0:
        return 'YES'
    else:
        return 'NO'


# solving
a, b = [int(num) for num in input().split(" ")]
r = solution(a, b)
print(r)
```

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИНФОРМАТИКА. ОТБОРОЧНЫЙ ТУР 10 КЛАСС

На уроке Алгебры

Кузьму заинтересовали последовательности и он определил понятия **ключ** и **сложность** последовательности.

Одно-сложной последовательностью $a_0,a_1,a_2,...$ Кузьма называет последовательность, **ключ** K которой содержит 1 элемент $[k_0]$ и элементы последовательности удовлетворяют формуле $a_i+1 = a_i + k_0$.

Примером одно-сложной последовательности является Арифметическая прогрессия.

М-сложной последовательностью $a_0,a_1,a_2,...$ Кузьма называет последовательность, **ключ** K которой содержит M элементов $[k_0,k_1,...,k_{m-1}]$ и элементы последовательности удовлетворяют формуле $a_1 = a_0 + k_0$, $a_2 = a_1 + k_1$, ..., $a_{i+1} = a_i + k_{i'}$, где $i`$ равен остатку от деления i / m.

Кузьма загадал возрастающую последовательность неизвестной сложности и просит Вас подобрать к ней **ключ**!

Формат ввода

В первой строке содержится число n ($4 \le n \le 1000$) - кол-во известных элементов последовательности.

Во второй строке содержится n чисел a_i ($1 \le a_{i-1} \le a_i \le 10^6$) - элементы последовательности.

Гарантируется, что кол-во известных элементов $n \ge 3 * m + 1$ (то есть ключ применен не менее трех раз)

Формат вывода

В ответ выведите элементы **ключа** k_i подходящего последовательности (если существует несколько подходящих **ключей**, выведите минимальный по длине)

Пример 1

Ввод	Вывод
10 1 2 4 5 7 8 10 11 13 14	[1, 2]

Пример 2

Ввод	Вывод
10 5 6 14 15 16 24 25 26 34 35	[1, 8, 1]

Пример решения

```
def solution(num, arr):
    delta = []
    diff = set()
    for x in range(num - 1):
        d = arr[x + 1] - arr[x]
        diff.add(d)
        delta.append(d)
```

Москва
2024-2025 г.

```
    if len(diff) == 1:
        return [delta[0]]
    else:
        result = []
        for x in range(2, len(delta) // 3 + 1):
            # ключ подходит[0], длина [1], текущий [2], сравнение [3]
            result.append([1, x, [], set()])

        for x in delta:
            for y in range(len(result)):
                if result[y][0] == 1:
                    result[y][2].append(x)
                    if len(result[y][2]) == result[y][1]:
                        result[y][3].add(' '.join([str(s) for s in result[y]
[2]]))
                        result[y][2] = []
                        if len(result[y][3]) > 1:
                            result[y][0] = -1

        for y in range(len(result)):
            if result[y][0] == 1:
                answer = []
                for z in range(result[y][1]):
                    answer.append(delta[z])
                return answer

    return -1


# solving
n = int(input())
a = [int(num) for num in input().split(" ")]
r = solution(n, a)
print(r)
```

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИНФОРМАТИКА. ОТБОРОЧНЫЙ ТУР 10 КЛАСС

На уроке Истории

Недавно домашним заданием Кузьмы стала парная подготовка доклада. Кузьме и напарнику необходимо было выбрать столетие и описать важнейшие события, произошедшие за этот период. Напарник Кузьмы определился со столетием и подготовил свою часть доклада, после чего передал Кузьме копию своей работы.

По неосторожности Кузьма уронил полученную копию в шредер (измельчитель бумаги) и теперь не знает, о каком столетии ему необходимо дописать доклад. До напарника он не дозвонился и теперь ему предстоит склеить измельченные листы бумаги, чтобы попытаться понять - о каком же столетии необходимо подготовить доклад.

Формат ввода

В первой строке содержится целое число n ($2 \le n \le 1000$) - кол-во измельченных кусочков дат.

Во второй строке содержится n строк s_i ($1 \le len(s) \le 4$) разделенных пробелом и состоящих только из цифр

Гарантируется, что из полученных строк можно собрать хотя бы один набор дат удовлетворяющий условиям:

- каждая дата D_i лежит в диапазоне ($100 \le D_i \le 1999$)
- целочисленный результат деления D_i / 100 одинаковый и равен Y - выбранное столетие

Формат вывода

Выведите Y - выбранное столетие, если оно определяется однозначно

Иначе выведите: UNKNOWN

Пример 1

Ввод	Вывод
5 1 5 74 179 8	17

Пример 2

Ввод	Вывод
7 1 7 8 8 9 78 1	UNKNOWN

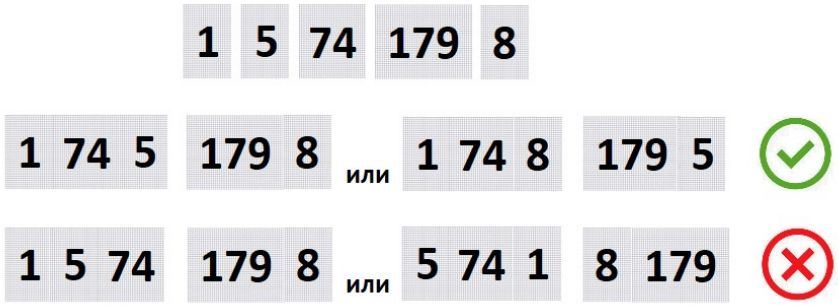
Примечания

В первом примере возможно собрать 1-74-5=**1745** и 179-8=**1798** (или 1-74-8=**1748** и 179-5=**1795**)

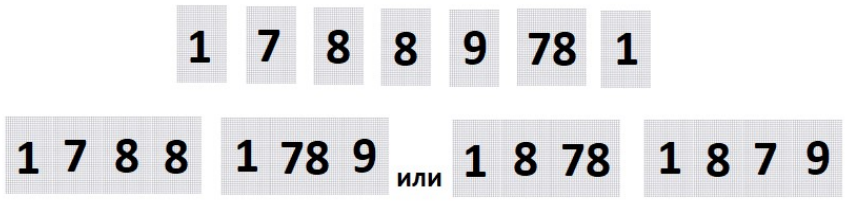
При этом собирать 1-5-74=**1574** и 179-8=**1798** нельзя, так как гарантируется, что даты обязаны принадлежать одному столетию

Также нельзя собирать 5-74-1=**5741** и 8-179=**8179**, так как даты не принадлежат заявленному диапазону

Таким образом ответ определяется однозначно: 17 (столетие)



Во втором примере можно по правилам собрать 1-7-8-8=**1788** и 1-78-9=**1789**, но можно и 1-8-78=**1878** и 1-8-9-7=**1897** - из-за чего столетие невозможно однозначно определить



Пример решения

```
def solution(num, dates):
    s = 0
    solver = []
    for x in range(10):
        # [x0], [x1] ... [x9] + свободные [x]
        solver.append([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

    date_max = '1'
    for x in range(num):
        # сумма цифр
        s += len(dates[x])
        # максимальный кусочек
        if len(date_max) < len(dates[x]):
            date_max = dates[x]
        # массив решений
        if len(dates[x]) == 1:
            solver[int(dates[x])][-1] += 1
        else:
            solver[int(dates[x][0])][int(dates[x][1])] += 1

    result = []
    for x in range(1, 20):
        if x < 10:
            if s % 3 == 0:
                count_dates = s // 3
                if len(date_max) == 3:
                    if date_max[0] == str(x):
                        if sum(solver[x]) >= count_dates:
                            result.append(x)
                elif len(date_max) < 3:
```

```
                    if sum(solver[x]) >= count_dates:
                        result.append(x)
        else:
            if s % 4 == 0:
                count_dates = s // 4
                if len(date_max) == 4:
                    return str(int(date_max) // 100)
                elif len(date_max) < 4:
                    x1 = x // 10
                    x2 = x % 10
                    if x1 != x2:
                        if solver[x1][x2] + min(solver[x1][-1],
sum(solver[x2])) >= count_dates:
                            result.append(x)
                    else:
                        sol1 = solver[x1][x1] + solver[x1][-1]
                        sol2 = solver[x1][x1] * 2 + (sum(solver[x1]) -
solver[x1][x1])
                        if sol1 >= count_dates and sol2 >= 2 * count_dates:
                            result.append(x)

    if len(result) == 1:
        return str(result[0])
    else:
        return 'UNKNOWN'


# solving
n = int(input())
d = input().split(" ")
r = solution(n, d)
print(r)
```