Задача 1. Стена

Один метр стены всегда имеет следующие размеры: ровно 4 кирпича в длину и 20 кирпичей в высоту. В самом нижнем слое половинчатых кирпичей не будет, а в следующем слое два половинчатых кирпича, т.е. там на один целый кирпич меньше. $10 * H$ рядов по $4 * L$ кирпичей, и $10 * H$ рядов по $4 * L - 1$ кирпичей. Итоговая формула $10 * H * 4 * L + 10 * H * (4 * L - 1)$. После упрощения получаем $80 * H * L - 10 * H$.

Задача 2. Правда или Ложь

Ответим на вопросы по порядку.

Во-первых, отметим, что правдолюбы из Правдино всегда отвечают «Да» на первый вопрос, лжецы из Лжецово также дадут положительный ответ, поскольку всегда говорят ложь. Точно так же хитрецы из Хитрецово солгут о своём местоположении и ответят «Д». Поскольку на первый вопрос было получено 360 ответов «Да», это означает, что во всех трёх населённых пунктах проживает 360 человек. Таким образом, это и есть ответ на первый вопрос.

На второй и третий вопросы было получено 480 ответов «Нет». Обратим внимание на то, что одновременно на оба вопроса отрицательно ответят только жители Правдино, так как остальные ответят «Да» про свой населённый пункт. Следовательно, из этих 480 ответов можно вычесть общее число жителей всех трёх поселений, чтобы получить количество жителей Правдино: $480 - 360 = 120$. Это ответ на второй вопрос.

Таким образом, количество жителей Лжецово и Хитрецово составляет $360 - 120 = 240$, что является ответом на третий вопрос.

Теперь рассмотрим следующее. Предположим, что оба утверждения осведомителя истинны. Тогда в Лжецово проживает в два раза больше людей, чем в Хитрецово. В этом случае количество лжецов будет равно $240 : 3 \cdot 2 = 160$, что является ответом на четвёртый вопрос.

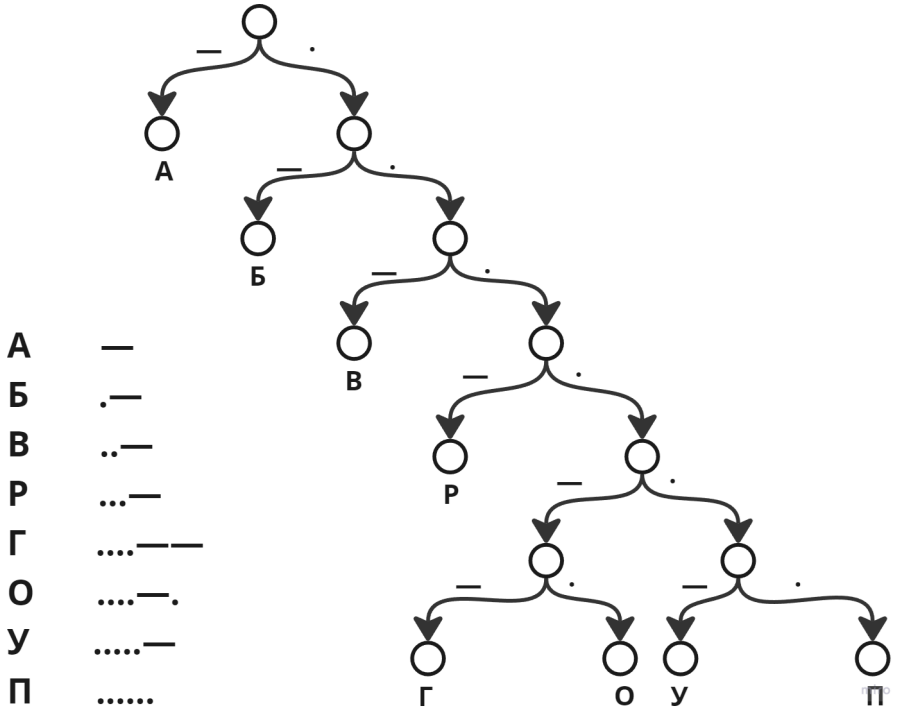
Если же предположить, что оба утверждения осведомителя ложны, то в Лжецово и Хитрецово живёт одинаковое количество людей и их число равно $240 : 2 = 120$. Это и есть ответ на пятый вопрос.

Задача 3. Азбука Морзе v2

Чтобы помочь Ване закодировать текст с минимальным количеством точек и тире, необходимо создать такой код, в котором более частым буквам соответствуют более короткие кодовые последовательности, а менее частым — более длинные. Это позволит сократить общее количество символов, необходимых для кодирования всего текста.

Сначала присвоим самые короткие кодовые последовательности наиболее частым буквам. Букве «А», встречающейся 100 раз, назначим самую короткую последовательность — из одного символа. Поскольку в азбуке Морзе используются точки и тире, для определённости выберем тире для обозначения «А». Букве «Б» с частотой 90 (второй по частотности) назначим последовательность из двух символов: он будет начинаться с точки (чтобы отличаться от кода «А») и заканчиваться тире.

Теперь необходимо присвоить коды остальным буквам так, чтобы ни одна кодовая последовательность не была началом другой и при этом общее количество символов было минимальным. Один из возможных вариантов представлен на следующем рисунке:

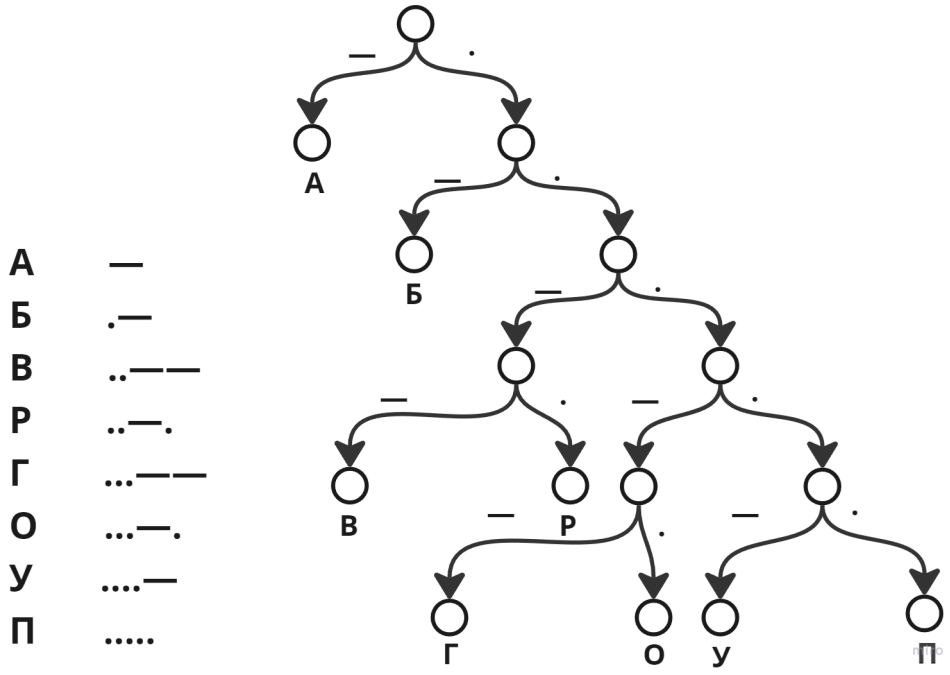


Здесь все коды расположены в виде дерева, где на линиях, соединяющих вершины, указаны точки или тире, а в листьях — буквы. Спускаясь от самой верхней вершины дерева к каждой букве, можно получить её уникальный код.

Теперь посчитаем общее количество символов, необходимых для кодирования текста:

$$100 \cdot 1 + 90 \cdot 2 + 25 \cdot 3 + 11 \cdot 6 + 10 \cdot 6 + 5 \cdot 6 + 9 \cdot 6 + 15 \cdot 4 = 625$$

Подумаем, как можно уменьшить это количество. Изменим коды букв «В» на `..-` -, а «Р» на `..-.` Для буквы «В» длина кодовой последовательности увеличится до 4 символов, а для «Р» останется прежней. Пересчитаем общее количество символов. Поскольку кодовая последовательность буквы «В» стала длиннее на один символ, общее количество символов увеличилось на 25, составив 650. На первый взгляд может показаться, что ситуация ухудшилась. Однако сейчас мы можем сократить длины последовательностей букв «Г», «О», «У», «И» с 6 до 5 символов, что уменьшит количество символов на $11 + 10 + 5 + 9 = 35$. В итоге получится 615 символов.



Анализ полученных кодовых последовательностей показывает, что это решение является наиболее оптимальным. В ответе следует записать соответствующие последовательности точек и тире.

Обратите внимание, что в ответе точки можно заменить на тире и наоборот, в итоге также получим верное решение.

Задача 4. Поезда

В таблице для каждой строки столбца B указано время прибытия поезда на станцию, а в столбце C — время его отправления. Скопируем значения из столбца B (кроме заголовка) на другой лист электронной таблицы в столбец A. У вас должно получиться 100 строк. В каждой строке с 1 по 100 в столбец B впишем число 1. Затем скопируем данные из столбца C (также без заголовка) в столбец A, начиная со строки 101, и для каждой строки с 101 по 200 запишем в столбец B число -1. Эти числа в столбце B показывают изменение количества поездов на станции: 1 — поезд прибыл (количество увеличилось на 1), -1 — поезд отправился (количество уменьшилось на 1).

Далее отсортируем строки по возрастанию значений в столбце A. Поскольку время в столбце A представлено в формате `ЧЧ:ММ`, где ЧЧ — часы, а ММ — минуты, сортировка упорядочит моменты времени по возрастанию.

Теперь добавим новую строку в начало таблицы и запишем в ячейку C1 значение 0. В столбце C будем хранить текущее количество поездов на станции для соответствующего времени в столбце A. Для этого в каждой строке со 2 по 201 нужно записать формулы, которые учитывают количество поездов в предыдущий момент времени и изменение количества поездов (из столбца B). Например, в ячейку C2 следует вписать формулу **=C1+B2** и затем скопировать её в диапазон C3:C201.

Обратите внимание, что в некоторых ячейках столбца C могут появиться отрицательные значения, что невозможно, так как количество поездов на станции не может быть отрицательным. Это связано с тем, что в ячейке C1 стоит значение 0, которое обозначает количество поездов на станции в полночь. Однако по условию задачи поезда уже могли находиться на станции до полуночи и отправляться после. Поэтому нужно подобрать минимальное значение для ячейки C1, при котором в столбце C не будет отрицательных значений. Это число 6.

Теперь можно приступить к ответам на вопросы задачи.

Для первого вопроса достаточно найти максимальное значение в столбце C. В ячейку D1 вписываем формулу **=МАКС(C1:C201)**, и результат — 10 — будет ответом на первый вопрос задачи.

Для второго вопроса нужно определить моменты времени, когда на станции находилось максимальное количество поездов. Таких моментов три: 05:02, 05:27 и 22:44. Максимальное количество поездов на станции будет сохраняться до следующего отправления какого-либо поезда. В нашем случае это 22, 2 и 12 минут соответственно. В сумме это составляет 36 минут — ответ на второй вопрос.

Для третьего вопроса поступаем аналогично, но ищем моменты, когда на станции не было ни одного поезда. Таких моментов два: 02:32 и 13:39. Время, в течение которого на станции не было поездов до моменты прибытия какого-либо поезда, составляет 18 и 11 минут соответственно. В сумме это 29 минут — ответ на третий вопрос.

Для ответа на четвёртый вопрос находим момент времени 10:00, и в соответствующей строке столбца B уже будет указан результат. В нашем случае это число 5.

Задача 5. Мастер-класс

Для получения 40 баллов можно было провести симуляцию процесса и вычислить длину маршрута.

Для полного решения, рассмотрим длину пути в случае, если длина стороны губки равна единице, а длина доски — n. Для перемещения в каждую из соседних клеток губка проходит длину 1. Так как клеток всего n^2, губка сделает $n^2 - 1$ шагов длиной 1.

А что происходит, когда длина губки не единичная? Этот случай мы можем свести к предыдущему, разделив длины сторон губки и доски на a, но увеличив длину одного перемещения в a раз. Итого губка сделает $(n/a)^2 - 1$ перемещений длины a. Итоговая формула выглядит как $((n/a)^2 - 1)a$.

Пример такого решения.

```
n = int(input())
```

```
a = int(input())
n //= a
squares = n ** 2
print(a * (squares - 1))
```

Задача 6. Наши слоны

Для получения 52 баллов достаточно явно перебрать все клетки шахматной доски и отметить подходящие.

Для получения полного балла необходимо рассмотреть несколько случаев.

1. $n = m$, и n чётно
2. $n = m$, и n нечётно
3. $n \neq m$

В первом случае левый верхний и правый нижний слоны «бьют» одну диагональ. Аналогично с правым верхним и левым нижним слоном. А также из чётности n следует, что эти диагонали не пересекаются. Итоговая формула $2(n-2)$.

Второй случай аналогичен первому, но диагонали пересекаются в одной клетке. Значит итоговая формула будет $2(n-2)-1$.

В третьем случае существуют 4 диагонали, по которым «бьются» клетки. Каждая из них независимо «бьёт» $min(n,\ m)-1$ клеток, однако надо рассмотреть потенциальные пересечения.

- Траектории движения верхнего левого и правого нижнего слонов не пересекаются; аналогично с другой парой слонов.
- Если траектории левого верхнего и правого верхнего слона пересекаются, то траектории левого нижнего и правого нижнего также пересекаются в другой точке; аналогично с другой комбинацией слонов.

Итого надо лишь проверить, пересекаются ли траектории левого верхнего и правого верхнего слонов, а также пересекаются ли траектории левого верхнего и левого нижнего.

Траектория левого верхнего слона пересекает траекторию правого верхнего, если:

- Ширина доски нечётна;
- Клетка пересечения находится в границах доски, а именно $(m+1)/2 \leqslant n$.

Если данные условия выполнились, то вычитаем из ответа 2. Аналогично проверяем для левого верхнего и левого нижнего.

Пример такого решения.

```
n, m = map(int, sys.stdin.read().split())
if n == m:
    if n % 2 == 1:
        print(2 * (n - 2) - 1)
    else:
        print(2 * (n - 2))
else:
    if n > m:
        n, m = m, n
    ans = 4 * (n - 1)
    if n % 2 == 1:
        ans -= 2
    if m % 2 == 1 and m <= 2 * n:
        ans -= 2
    print(ans)
```

,

Задача 7. Пирамидки

Чтобы получить 20 баллов, можно для каждой пирамидки в программе отдельно прописать, сколько в ней рядов, и из этого посчитать количество кубиков.

Чтобы получить 50 баллов, можно перебирать номер пирамидки в цикле и считать для каждой из них количество рядов и кубиков. Узнать количество рядов в пирамидке с номером i удастся, явно перебирая степень двойки, на которую делится i.

Чтобы получить полный балл, необходимо для каждого слоя посчитать, в скольких пирамидках он находится. При нумерации рядов начиная с первого i-й ряд будет присутствовать в $n//2^{i-1}$ рядах. Для простоты пронумеруем ряды с нулевого, тогда количество рядов, в которых будет i-й ряд, составит $n//2^i$, а в общую сумму i-й ряд привнесёт $(n//2^i)a_i$ кубиков. Номера рядов переберём в цикле.

Пример такого решения.

```
n = int(input())
k = int(input())
a = []
for i in range(k):
    a.append(int(input()))
p = n
ans = 0
j = 0
b = 1
while (p > 0):
    ans += p * a[j]
    b *= 2
    p = n // b
    j += 1
print(ans)
```