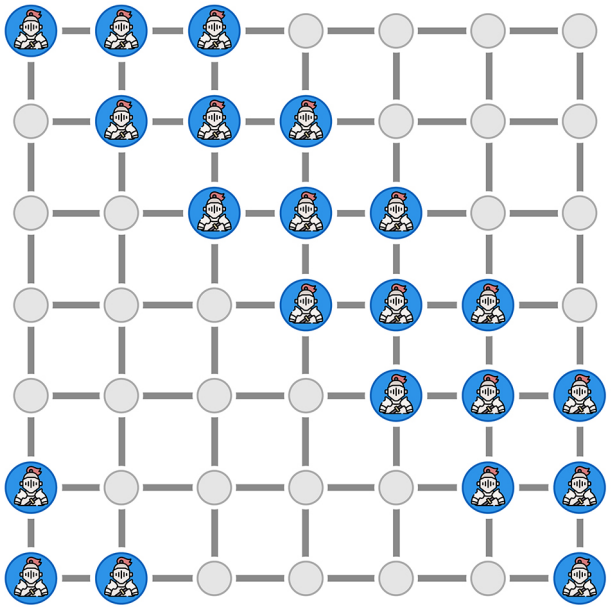
Задача 1. Стражники

Поскольку для каждой из 7 улиц одного направления требуется минимум три стражника, то всего их должно быть не менее 21. Можно обойтись ровно 21 стражником, если разместить трёх стражников в начале первой строки, а каждую следующую строку получать циклическим сдвигом предыдущей вправо на одну позицию.



Задача 2. Квадрат

Получим искомый квадрат, отрезав четыре прямоугольных треугольника с катетами a/n и $a(n-1)/n$ от исходного квадрата со стороной a. Два таких треугольника в сумме дают один прямоугольник со сторонами a/n и $a(n-1)/n$, поэтому площадь отрезанных треугольников равна $2a^2(n-1)/n^2$.

Ответ можно записать как $a*a-2*a*a*(n-1)/(n*n)$.

Знакомые с теоремой Пифагора также могут заметить, что гипотенуза прямоугольного треугольника с катетами a/n и $a(n-1)/n$ как раз является стороной закрашенного квадрата. Поскольку площадь искомой фигуры равна квадрату её стороне, т.е. сумме квадратов катетов a/n и $a(n-1)/n$, это приводит к ответу $a*a/(n*n)+a*a*(n-1)*(n-1)/(n*n)$. Такая форма записи ответа, а также любое другое эквивалентное выражение, тоже будут правильными.

Задача 3. Склад

Можно решить задачу за 6 минут, выполняя каждую минуту ровно два перемещения. Пример такого решения (пустые строки добавлены для удобства):

1 4
2 3

1 2
3 4

3 1
4 2

4 1
3 2

2 1
4 3

1 4
2 3

Задача 4. Ремонт дороги

Предполагается решать эту задачу с использованием приложения для работы с электронными таблицами.

Отсортируем таблицу по номеру начального участка каждого подрядчика, то есть по столбцу B. Нам необходимо выбрать первую компанию, которая отремонтирует начало дороги, т.е. участок начиная от города 1. После сортировки такие подрядчики окажутся в строках 2:10 таблицы. Выберем из них компанию с максимальным номером города в конце участка (максимальным значением в столбце C). Это подрядчик номер 62, который может отремонтировать отрезок $[1; 169]$. Всегда можно построить ответ с использованием именно этой компании, потому что любого другого подрядчика с началом участка в городе 1 и меньшим номером конца можно заменить на компанию, также начинающую от города 1, но заканчивающую в населенном пункте с большим номером.

Выберем следующую компанию, которая продолжит ремонт начиная от города 169, то есть левый конец её отрезка должен быть не больше 169. В отсортированной таблице эти компании будут располагаться в строках до 33-й. Из аналогичных соображений нужно выбрать исполнителя, способного отремонтировать участок до города с максимальным номером. Этому условию отвечают подрядчики номер 79 с отрезком $[163; 275]$ и номер 88 с отрезком $[154; 275]$. Можно выбрать любого из них.

Продолжаем процесс дальше, находим в отсортированном списке компаний те, номер города в начале участка которых не превосходит 275, и выбираем исполнителя с максимальным значением на конце отрезка: номер 6 с отрезком $[240; 371]$.

Аналогично добавим компании номер 58 (отрезок $[371, 460]$), номер 21 (отрезок $[442, 617]$), номер 55 (отрезок $[608, 735]$), номер 76 (отрезок $[726, 876]$), номер 82 (отрезок $[870, 993]$), затем можно выбрать любую компанию, завершающую участок в точке 1000 (кроме компании номер 4), например, подрядчика номер 68 (отрезок $[902, 1000]$).

Ответ: 62, 79, 6, 58, 21, 55, 76, 82, 68. Есть и другие ответы, также содержащие девять компаний.

Задача 5. Лягушка и кузнечик

Расстояние между лягушкой и кузнечиком равно $N - 1$ клеток. За одну секунду это расстояние может сократиться на 3, 4 или 5 клеток. Поэтому если $N - 1 = 1$ или $N - 1 = 2$ (то есть при $N < 4$) лягушка и кузнечик не смогут встретиться и нужно вывести «-1».

Во всех остальных случаях им понадобится $\lceil \frac{N-1}{5} \rceil$ прыжков (частное от деления $N - 1$ на 5, округлённое вверх). Это значение можно вычислить по формуле `(N - 1 + 4) // 5` (где `//` — операция целочисленного деления в языке Python, в языках C++, C#, Java надо использовать просто `/`, в языке Pascal — операцию `div`), или при помощи действительного деления и округления результата вверх при помощи функции `ceil`.

Пример решения на языке Python.

```
n = int(input())
if n < 4:
    print(-1)
else:
    print((n - 1 + 4) // 5)
```

Если решать задачу «моделированием», например, в цикле увеличивать координату лягушки на 3, а координату кузнечика уменьшать на 2, то такое решение наберёт 50 баллов.

Кроме того, частичные решения могут быть основаны на идее перебора. Например, можно перебирать величины f_2, f_3, g_1, g_2 — количество прыжков лягушки на 2 и на 3 клетки и количество

прыжков кузнечика на 1 и на 2 клетки соответственно — и в качестве кандидатов на оптимальное решение рассматривать такие варианты, в которых

$$2f_2 + 3f_3 + g_1 + 2g_2 = N - 1$$

и

$$f_2 + f_3 = g_1 + g_2.$$

В этом случае получится очень медленное решение, имеющее вычислительную сложность $O(N^4)$ и набирающее 30 баллов.

Также допустимо перебирать количество секунд s с момента старта и анализировать, могут ли лягушка и кузнечик оказаться в одной клетке по прошествии данного времени. Потребуется выяснить, пересекаются ли множества клеток, в которых они могут оказаться. Нетрудно понять, что для лягушки такое множество — это отрезок полосы с номерами клеток от $1 + 2s$ до $1 + 3s$ включительно, а для кузнечика — отрезок с номерами клеток от $N - 2s$ до $N - s$ включительно. Таким образом, задача сводится к нахождению такого минимального s, при котором данные отрезки пересекаются. Если этот поиск осуществлять последовательным перебором, решение будет иметь вычислительную сложность $O(N)$ и набирать 50 баллов. Также минимальное подходящее s можно найти методом двоичного поиска, что даёт вычислительную сложность $O(\log N)$ и будет оцениваться полным баллом.

Задача 6. Устный опрос

Если значение K — нечётное, то Вася окажется в первой половине опрошенных, поэтому ответом будет порядковый номер числа K среди всех нечётных чисел. Его можно найти по формуле `(K + 1) // 2` (где `//` — операция целочисленного деления).

Если K — чётное, то нужно определить, какое это по порядку чётное число, а также добавить количество нечётных чисел от 1 до N. В этом случае получится формула `K // 2 + (N + 1) // 2`.

Пример решения на языке Python.

```
n = int(input())
k = int(input())
if k % 2 != 0:
    print((k + 1) // 2)
else:
    print(k // 2 + (n + 1) // 2)
```

50 баллов можно набрать, если составить список всех учащихся в том порядке, в котором они выходят, то есть сначала из нечётных чисел, потом из чётных чисел, а затем найти в этом списке позицию элемента K.

```
n = int(input())
k = int(input())
a = [i for i in range(1, n + 1, 2)] + [i for i in range(2, n + 1, 2)]
print(a.index(k) + 1)
```

Задача 7. Заказ в магазине

Сначала проверим, существует ли решение. Максимальное количество ручек, которое можно заказать, равно $1 + 2 + 3 + ... + N = N(N + 1)/2$ (по формуле суммы арифметической прогрессии), и если $M > N(N + 1)/2$, то решения не существует.

При проверке этого условия можно столкнуться с проблемой переполнения 32-битного целого типа в языках C++, Pascal, Java, C#, поэтому вычисление нужно производить с использованием 64-битных целочисленных переменных. Также если эту сумму вычислять не по формуле, а при помощи цикла, то цикл длины N можеть не уложиться в ограничение по времени. В этом случае стоит прервать цикл, если сумма превысит M, либо заметить, что если $N > \sqrt{2 \cdot 10^9} \approx 44721$, то $1 + 2 + 3 + ... + N > 10^9$ и решение существует.

Если решение существует, то воспользуемся жадным алгоритмом: будем выбирать упаковки максимально возможного размера: N, $N-1$, $N-2$ и т.д. Если размер рассматриваемой упаковки s больше или равен M, то выведем значение s и уменьшим M на s.

Если использовать цикл `for` от N до 1, то получится решение сложности $O(N)$, которое наберёт 40 баллов. Надо заметить, что вовсе необязательно перебирать все значения от N до 1, т.к. если в какой-то момент значение M станет меньше рассматриваемого размера упаковки, то достаточно взять одну упаковку размером M, то есть вывести M и завершить работу программы. Такое решение будет иметь сложность $O(\sqrt{M})$.

Пример решения на языке Python.

```
n = int(input())
m = int(input())
if (1 + n) * n // 2 < m :
    print(0)
else:
    while m > n:
        print(n)
        m -= n
        n -= 1
    print(m)
```