Задача 1. 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 2. 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 3. 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)
```

Задача 4. Тайна магического домино

Для начала рассмотрим идею решения, которое набирает не менее 50 баллов. Учитывая, что количество костяшек не превышает 1000, можно воспользоваться алгоритмом с асимптотикой $O(n^2)$. Заведём массив $fallen[n]$, в котором будем отмечать, упала ли i-я костяшка. Для этого начнём идти слева направо, и если про i-ю костяшку мы знаем, что она упадёт, то будем идти направо $i \leqslant j \leqslant n$ и помечать j-ю костяшку как упавшую, если разность их координат меньше или равна высоте i-й костяшки. Аналогично справа налево.

Однако задачу можно решить и за линейное время. Для этого будем поддерживать $border = max(border, x[j] + h[j])\ \forall j :\ 0 \leqslant j \leqslant i$, где i - текущая костяшка. Проще говоря, поддерживать на каждом шаге такую максимальную координату, что костяшки на координатах меньше данной будут задеты падающими «соседками». А далее будем отмечать i-ю костяшку как упавшую, и пересчитывать $border$, только если $border \geqslant x[i]$. Таким образом, за линейное время можно отметить все костяшки, которые упадут от импульса. Ситуация при костяшках, падающих справа налево, обрабатывается аналогично.

Пример кода, набирающего полный балл.

```
n = int(input())
x = [int(input()) for i in range(n)]
h = [int(input()) for i in range(n)]
border = x[0]
i = 0
while i < n and x[i] <= border:
    border = max(border, x[i] + h[i])
    i += 1
j = n - 1
border = x[n - 1]
while j >= 0 and x[j] >= border:
    border = min(border, x[j] - h[j])
    j -= 1
print(min(n, i + (n - 1 - j)))
```

Задача 5. Покраска забора

Для решения при $n <= 10$ можно было перебрать все возможные варианты выбора банок с краской.

Далее рассмотрим идею полного решения. Для начала научимся считать ответ, если мы знаем, что будем использовать банки ёмкостью $b_1, b_2, \ldots, b_m$ (именно в таком порядке). Также для простоты будем считать, что $b_1 + \ldots + b_m = k$. Мимо первых b_1 досок Егор пройдёт $2 \cdot m - 1$ раз, потому что будет проходить мимо них по 2 раза с каждой банкой (не считая последней). Мимо следующих b_2 досок Егор пройдёт $2 \cdot m - 3$ раз и так далее. Также потребуется ещё k секунд, чтобы покрасить весь забор. Значит, итоговое время составит $k + \sum\limits_{i=1}^{m}(2m - 2i + 1) \cdot b_i$ секунд. Преобразуем это выражение: $k + \sum\limits_{i=1}^{m}(2m - 2i + 1) \cdot b_i = k + (2m + 1)\sum\limits_{i=1}^{m} b_i - 2\sum\limits_{i=1}^{m} b_i \cdot i$. Можно видеть, что тут есть три части, но первые две не меняются от перестановки b_i, поэтому нас интересует только последняя. Чтобы вычесть как можно больше, необходимо отсортировать b_i по возрастанию. Значит, для получения наименьшего времени, все b_i должны идти в порядке возрастания.

Также несложно видеть, что выгодно взять самые большие банки с краской так, чтобы суммарное количество краски в них было не меньше k. Это выгодно, так как при другом ответе можно взять какую-то банку с большим количеством краски и это точно будет не хуже.

Итоговое решение: отсортируем все банки. Пока забор не закончился, будем брать банку с максимальной ёмкостью из оставшихся и красить забор настолько, насколько можем.

Пример решения:

```
import sys

k = int(input())
n = int(input())
v = [int(input()) for i in range(n)]
if sum(v) < k:
    print(-1)
```

```
    sys.exit(0)
ans = 2 * k
v.sort()
while k > 0:
    k -= v[-1]
    v.pop()
    if k > 0:
        ans += 2 * k
print(ans)
```