Задача A. Стакан

В этой задаче требовалось просто сложить три целых числа.

Задача B. Четыре чертенка

Проверить, что два каких-то введенных числа равны, можно при помощи довольно громоздкого `if`. Но можно подумать, как упростить условие для проверки, если единица, двойка и тройка гарантированно встретятся среди введенных чисел.

Зафиксируем мысленно трех чертят, которые заняты разными частями чертежа. Посмотрим на действия последнего (четвертого) чертенка. Если он ничего не делает, то ответ — «`NO`». Если же он чертит любую из трех частей, то он это делает совместно с одним их трех зафиксированных чертят, тогда ответ — «`YES`».

Таким образом, ответ «`NO`» достигается только в том случае, если среди введенных чисел не встречается ноль. А так как все числа по условию неотрицательные, условие можно упростить до: `if min(a1, a2, a3, a4) == 0`.

Задача C. Пицца

Пусть Боря приведет N друзей. Чтобы всем досталось поровну, должно существовать такое d — количество кусков для каждого из пришедших, что $N \cdot d = K$. Тогда K должно делиться на N. Значит, нужно при помощи цикла найти максимальное нечетное число N, которое делит K.

Бонус: Как решать эту задачу при $K \leqslant 10^{18}$?

Задача D. Мем ли?

Задача состоит из двух частей.

Первая: посчитать количество заглавных и строчных букв. Это можно сделать при помощи прохода циклом `for` по строке с `if` внутри цикла. Отметим, что для проверки регистра необязательно было использовать встроенные функции или сравнивать коды символов. Вместо этого можно было написать длинный `if`, перебирающий все возможные варианты заглавных букв, которых по условию всего четыре: «L», «O», «K» и «E».

Вторая: проверить условие. Для этого следует воспользоваться функцией взятия модуля числа. Пусть a — количество заглавных букв, а b — строчных. Тогда достаточно проверить, что $|a - b| \leqslant 1$.

Задача E. Убург

Подсказка 1: перенумеруйте дома от 1 до 12.

Подсказка 2: даже на маленьких числах имеет смысл написать `for`.

Заметим, что вам даны часы, циферблат которых разбит на 3 сектора. Действительно, можно перейти от задачи с городом к задаче про циферблат, если определить час на старте как $c = 4 \cdot (a-1)+x$ и час на финише как $d = 4 \cdot (b-1) + y$. Теперь требуется посчитать, сколько цифр пройдет стрелка от часа c к часу d.

Такую задачу можно решить двумя способами. Первый — смоделировать движение стрелки при помощи цикла. Для этого просто будем прибавлять на каждой итерации цикла единицу к c, если $c < 12$, а иначе менять значение c с 12 на 1. Таким образом, c будет показывать в цикле час, на который в текущий момент показывает стрелка часов. Для ответа нужно также держать счетчик, через сколько цифр уже прошла стрелка, и на каждой итерации прибавлять к нему единицу.

Другой подход — заметить, что если $d > c$, то ответ на задачу: $d-c+1$. Тогда если c и d поменять местами, то будет посещена оставшаяся часть циферблата, а также стартовые и конечные точки. Поэтому если $d < c$, то ответ равен $n - (d - c + 1) + 2 = n - d + c + 1$.

Задача F. Упрощенный боулинг

Подсказка 1: Можно ли решить эту задачу вообще без десяток?

Подсказка 2: В задаче два случая с $k > 90$ и $k \geqslant 90$.

Постараемся избавиться от особенности начисления очков при десяти кеглях.

Можно никогда не ставить десятки. Для этого можно явно ограничить каждое число девятью, и выводить минимум из 9 и оставшихся нераспределенных очков, которые нужно добавить в протокол. Таким методом можно получить до $9 \cdot 10 = 90$ очков, что для решения задачи недостаточно.

С другой стороны, когда $k > 90$ можно поставить несколько десяток, а в оставшиеся клетки расставить девятки. Например, начнем с четырех десяток и девятки. Тогда сумма за первые пять раундов будет 88. Осталось за пять раундов добрать еще очков, используя числа, не больше девяти. Можно получить таким образом все результаты до 133, что нам достаточно.

Бонус: Как решить задачу при $k \leqslant 190$? С такими ограничениями не для всех k гарантируется, что существует вообще какой-либо протокол, по которому результат — k баллов.

Задача G. Бамбук

Посмотрим на итоговые стебли. Зафиксируем стебель x, скорость роста которого изменена из-за натянутой веревки. Для определенности пусть веревка натянута слева от него. Тогда его длина равна длине стебля слева плюс d.

Давайте уберем все веревки и посмотрим, как могли бы расти все стебли без ограничений (тогда i-й стебель имел бы длину $b_i \cdot t$). Затем начнем навешивать веревки и добавлять ограничения.

Понятно, что навесить веревку на x-й стебель можно простой операцией: $len[x] = \min(len[x], len[x-1] + d)$, где $len[i]$ — длина i-го стебля (с учетом какого-то числа веревок). Однако это работает только если на длину стебля слева ничего не влияет. Но тогда заметим, что если обновить перед этой операцией длину $(x-1)$-го стебля, то длина посчитается верно. Обобщая, перед навешиванием веревки нужно, чтобы у всех соседей слева длина была посчитана верно.

Понятно, что если бы все веревки могли ограничивать рост только правого соседа, то этот пересчет бы работал, однако веревки двусторонние: они могут ограничить как рост соседа слева, так и справа. Но заметим, что если в итоге x-й стебель будет ограничивать стебель слева от него, то стебель слева не может быть ограничен в росте стеблем справа. И так для всей последовательности стеблей слева, которые сами ограничены ростом стеблем слева.

Таким образом, если пройтись циклом слева направо и обновить навесить «односторонние» веревки путем обновления длин стеблей $len[x] = \min(len[x], len[x-1] + d)$, длина всех стеблей, на чью итоговую длину повлияли соседи справа, будет корректно посчитана. Поэтому осталось только реализовать симметричный случай справа налево.