Задача A. Ультраотрезок

Для решения задачи нужно было понять единственный факт про ультраотрезок: единственное число, которое делит любое другое больше себя — это единица. Решения данной задачи сводилось к проверке l на равенство единице. Если равно, то это ультраотрезок, а иначе — нет.

Задача B. Стипендия

По условию, нам необходимо максимизировать *среднее арифметическое* стипендий — общую сумму стипендий, деленную на число стипендиатов. Всего стипендиатов ровно a, а общая сумма стипендий не более m. Следовательно, ответ не превосходит $\frac{m}{a}$.

Покажем, что всегда можно назначить стипендии согласно условию таким образом, чтобы суммарно выплачивать m дублей. Назначим базовую стипендию в размере $\frac{m}{a+b}$ дублей, а повышенную — в размере $\frac{2m}{a+b}$ дублей. Тогда суммарные выплаты по базовым стипендиям составят $(a-b)\cdot\frac{m}{a+b}$ дублей, по повышенным — $b\cdot\frac{2m}{a+b}$ дублей. В сумме получаем ровно m дублей.

Следовательно, в качестве ответа необходимо вывести значение $\frac{m}{a}$.

Задача C. Постулат Бер...на

В задаче описан постулат Бертрана, который звучит так: для любого натурального $n \geqslant 2$ в промежутке $(n, 2n)$ всегда найдется простое число. И да, он доказан. Следовательно всегда будет ответ «`YES`», кроме случая $n = 1$. В итоге осталось найти простое число в интервале. Можно просто пройтись циклом от $n + 1$ до $2n - 1$ и проверить каждое число на простоту. Это решение будет работать быстро, поскольку интервалы между соседними простыми числами небольшие (см. `http://gg.gg/fv6dk`).

Если проверять числа на простоту перебором всех потенциальных делителей от 2 до $n - 1$ и проверкой остатка от деления, то решение будет работать за линейное время и наберёт 60 баллов. Однако, если $d > \sqrt{n}$ является делителем n, то и $\frac{n}{d}$ будет делителем n, следовательно, перебирать делители можно до $\sqrt{n}$. Решение с учётом этого набирает полный балл.

Задача D. Лестница

Рассмотрим две соседние ступеньки, пусть их высота будет p и q $(p < q)$. Если $q - p \leqslant x$, то дополнительные ступеньки между ними не нужны. Если $x < q - p \leqslant 2x$, то нужно вставить одну ступеньку. Если $2x < q - p \leqslant 3x$, то необходимо две ступеньки, и так далее. Таким образом, число необходимых ступенек можно представить как $\frac{q-p-1}{x}$.

Тогда решение может быть таким: пройдемся по массиву и найдем сумму величин $\frac{a_{i+1}-a_i-1}{x}$ для всех пар соседних ступенек.

Поскольку на каждом шаге необходимо рассматривать только текущее и предыдущее значение, можно решать эту задачу без использования массивов.

Задача E. Факториал

Примитивное решение включает в себя наивный подсчёт факториала, удаление нулей и взятие остатка по модулю 10. Однако, такое решение работает некорректно, поскольку при умножении, например, на 15, появится третий разряд, и последняя цифра будет вычислена некорректно, поскольку мы не храним большие числа.

Однако, поскольку ограничения невелики, мы можем хранить остаток по модулю $10^{\log_5 n} \approx 15\,000\,000$, тогда «переполнения» не будет.

Есть и другой путь решения, который не использует подбор констант: мы можем вычислять факториал по модулю 10, при этом не умножать на двойки и пятёрки (например, вместо числа 20 мы умножим на 1, а вместо 24 — на 3). Само же количество пропущенных двоек и пятёрок сохраним в переменные (можно хранить одну переменную, содержащую разность этих величин). Поскольку $2 \times 5 = 10$, мы можем убрать некоторое число «лишних» пятёрок и двоек, пока у нас не закончатся пятёрки, а дальше умножить полученное число на все оставшиеся двойки. В разложении нет ни одной пятёрки, поэтому последняя цифра точно не будет нулём.

25 баллов можно было набрать, посчитав факториал полностью, и потом убрав нули, в 64-битных типах данных на языках C++, Pascal. 40 баллов набирали решения на Python, делающие то же самое, используя встроенные длинные числа.

Задача F. Рунология

11 баллов можно было получить достаточно просто, поскольку лишних символов в строке нет. Значит, s состоит только из анаграмм строк t_i. Тогда длина каждого куска строки будет равна длине соответствующей руны, и осталось лишь подсчитать границы отрезков. Будем запоминать длины уже просмотренных рун. Тогда $l_{i+1} = r_i + 1$ — кусок начинается со следующего символа, а $r_i = l_i + len(t_i) - 1$ — длина куска будет равна длине руны.

Более оптимальное решение включает в себя аккуратную реализацию. Рассмотрим текущую руну. Для каждого символа найдем то место, где он встречается впервые в нашей строке. Пометим этот символ как используемый (авторское решение заменяет букву на символ «-»). Как только мы нашли все символы руны, можно завершать кусок и переходить к следующей руне. Однако, данное решение имеет квадратичную сложность и набирает 90 баллов.

Чтобы получить максимальный балл, необходимо для каждой руны запомнить, сколько определенных символов встречалось в руне (например, для руны «`abacaba`» используется четыре буквы «`a`», две буквы «`b`» и одна буква «`c`»). После этого начнём проходить по строке, и, если текущая буква требуется для руны, уменьшим число оставшихся требуемых букв. Если мы нашли все буквы, опять же, можно переходить к следующей руне. Это решение работает за линейное время и набирает полный балл.