Задача A. Яичница

Для решения этой задачи требуется вывести $X - Y$.

Однако в этой задаче есть две вспомогательные группы тестов, позволяющие решать ее поэтапно.

Для прохождения 1-й группы тестов можно не считывать вводимые числа вовсе. Достаточно написать вывод числа 7 — ответа на задачу при $X = 7, Y = 0$.

Во 2-й группе тестов $Y = 0$, поэтому ответ равен $X - 0 = X$. Таким образом, для прохождения второй группы тестов достаточно считать только число X и вывести его.

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		X	Y	
1	17	$X = 7$	$Y = 0$	—
2	34	$X \leqslant 50$	$Y = 0$	1
3	49	$X \leqslant 50$	$Y < X$	1, 2

Баллы выставляются автоматически проверяющей системой.

Задача B. Удвоение

Решение задачи предполагает разбор всех случаев. Для этого требуется сравнить D при помощи условных операторов с двумя числами: $2A$ и $2B$. Если D равно хотя бы одному из них, то ответ — «`YES`», иначе — «`NO`».

В первых двух группах тестов $D = 2$, поэтому ответ «`YES`» достигается, только если хотя бы одно из чисел A или B равно единице. При этом для прохождения первой группы достаточно сравнить только число A с единицей, так как $A = B$.

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		D	Дополнительно	
1	25	$D = 2$	$A = B$	—
2	52	$D = 2$	—	1
3	23	$D \leqslant 200$	—	1, 2

Баллы выставляются автоматически проверяющей системой.

Задача C. Забор

Для решения задачи необходимо написать алгоритм, генерирующий подходящую строку. Примером такой строки является: «`queryqueryquery...`» (всего 100 букв).

Но как такую строку придумать? Начнем поэтапно решать задачу.

Чтобы пройти первые 3 теста можно попробовать дописать несколько букв к строке из примера. Многие строки подошли бы. При этом можно убедиться в корректности строки как ручной проверкой на листочке, так и просто путем отправки придуманного слова на проверку.

Изучим какую-нибудь хорошую строку, например, «`dogabc`». Почему эта строка подходит? Первое наблюдение: для того, чтобы в строке не появлялись палиндромы (будем считать, что по умолчанию имеются ввиду палиндромы неединичной длины) достаточно, чтобы все буквы были различны. Действительно, если никакие буквы не совпадают, то все подстроки гарантированно читаются поразному слева направо и справа налево, например, из-за того, что первая и последняя буквы будут отличаться. Таким образом, слово, в котором все буквы различны, точно подходит.

Значит, для прохождения первых 5 тестов достаточно вывести весь латинский алфавит. Но попробуем приписать еще какую-то букву к алфавиту, и почти наверняка такое решение пройдет и 6-й тест. Действительно, даже если мы найдем подстроку, в которой первая и последняя буквы будут совпадать, необходимо чтобы совпали и некоторые другие пары букв, например, вторая и предпоследняя буквы.

Ключевая идея: если в строке найдена подстрока-палиндром длины l, то можно убрать первую и последнюю буквы в этой подстроке, и тогда полученная подстрока длины $l-2$ должна остаться палиндромом. Но так можно делать, пока от исходной подстроки не останется подстрока длины 2 или 3, которая должна быть палиндромом. Значит, чтобы в строке отсутствовали подстроки-палиндромы, достаточно только гарантировать, чтобы все подстроки длины 2 и 3 не являлись палиндромами.

Но тогда не имеют значения повторения символов, если они расположены хотя через два символа друг от друга. Поэтому достаточно придумать строку длины хотя бы 3, в которой все символы различны, а потом ее дублировать до тех пор, пока ее длина не будет равна 100 (при превышении лишние символы можно отбросить). Пример такой строки приведен в начале разбора.

Критерии оценивания:

№	Баллы за тест	N
1	11	$N = 4$
2	9	$N = 5$
3	5	$N = 6$
4	8	$N = 10$
5	17	$N = 26$
6	22	$N = 27$
7	11	$N = 30$
8	8	$N = 50$
9	9	$N = 100$

Баллы выставляются автоматически проверяющей системой

Задача D. Бодрые квартиры

Для прохождения 1-й группы тестов достаточно проверить, включен ли свет в единственном окне.

Во 2-й группе тестов в доме только одна квартира, но ее длина равна n. Поэтому требуется посчитать количество окон с выключенным светом во всем доме.

Такое решение можно реализовать при помощи прохода по строке циклом `for`. Для этого заведём вспомогательную переменную — количество окон, в которых свет не горит. К ней в теле цикла будем прибавлять единицу за каждое окно с выключенным светом, то есть за каждый элемент строки, равный символу «`.`». Отметим, что строку с цветом окон нужно считать, но при этом решение ее никак не использует.

Во многих языках программирования эта группа решается при помощи встроенных функций. Такую реализацию можно использовать, в том числе и как подзадачу для решения следующих групп. Однако далее будет рассматриваться именно реализация через цикл.

Отметим, что можно считать во вспомогательной переменной как количество окон, в которых не горит свет, так и число окон, в которых свет горит. В этом решении от этого зависит только условие, определяющее ответ: 0 или 1. В одном случае, нужно проверить, что во вспомогательной переменной после цикла лежит значение 0, а в другом — n.

Переходя к следующей группе, заметим, что дом с несколькими квартирами можно искусственно разбить на несколько домов с одной квартирой и посчитать ответ на задачу как сумму ответов по всем домам, на которые был разбит исходный дом.

Таким образом, для решения 3-й группы тестов следует отдельно рассмотреть случай, когда в доме две квартиры. Обнаружить, что в доме две квартиры (если гарантируется, что их не больше, чем две), можно сравнением цвета первого и последнего окна. Если эти цвета не отличаются, то квартира одна, и можно запустить решение для 2-й группы тестов.

В противном случае, найдем при помощи цикла или встроенной функции место, где начинается вторая квартира. Для этого можно найти самое левое окно, цвет которого отличается от цвета первого окна, то есть можно найти символ в строке цветов окон, не равный первому символу в этой строке — это будет первое окно второй квартиры. Далее можно явно разбить каждую из введенных строк на две по этому индексу, тем самым разбить исходный дом на два новых, а затем вставить решение второй группы тестов для каждого из них, объединив ответы при помощи суммы.

Отметим, что эту группу можно решить и не создавая новых строк, а, например, выделив два диапазона индексов для циклов по первой и второй части дома: от первого окна до места смены цвета окон и от места смены цвета окон до конца дома. Тело цикла при этом остается таким же, как и в решении второй группы тестов.

Идея с разбиением обобщается для любого числа квартир. Однако реализовать явное разбиение строки на несколько подобным образом довольно трудно. Поэтому посмотрим, как можно проводить такое разбиение неявно, то есть не создавая новых строк.

Опытные участники скорее всего знают, что подобное неявное разбиение можно реализовать при помощи обнуления вспомогательных переменных в тех местах, где разбиение происходит. Таким образом, первый вариант полного решения задачи состоит в следующем. Сначала нужно завести переменную, в которую суммируются ответы по каждой квартире (прибавляется 1, если в рассмотренной квартире во всех окнах горит свет), причем обновление ответа осуществляется через вспомогательную переменную, почти как в решении второй группы тестов, небольшие отличия обозначим позже. Однако обновление ответа должно запускаться не только в самом конце, но и при перемещении из одной квартиры в другую, то есть при смене цвета окон. Помимо этого нужно обнулять счётчик числа окон с включенным (или выключенным) светом, тем самым запускать подсчёт заново при перемещении из квартиры в квартиру.

Отличие условия обновления ответа касается только реализации, в которой во вспомогательной переменной считается количество окон с горящим светом — значение в счётчике нужно сравнивать не с n, а с длиной квартиры, которую можно посчитать, например, при помощи другой вспомогательной переменной. При этом условие, если счётчик считает количество окон с выключенным светом, остается прежним.

Если такая идея незнакома, то к похожему решению можно прийти через следующие группы тестов. При этом саму идею, что писать явное разбиение не следует, подсказывает 4-я группа тестов, в которой у всех квартир ровно одно окно. Но тогда количество квартир, в которых горит свет во всех окнах, в точности равно числу окон, в которых горит свет, а это то, что считает вспомогательная переменная в решении для 2-й группы (или то, что можно вывести простой формулой, если вспомогательная переменная считает число окон с выключенным светом). Таким образом, для решения 4-й группы тестов нужно только изменить вывод ответа в решении для 2-й группы.

Однако что в таком случае будет считать вспомогательная переменная? Очевидно, что это сам ответ на задачу. Тогда кажется, что раз старый счётчик больше не участвует в реализации, то и вообще как таковой подсчет количества окон с включенным светом отсутствует. Но он, конечно, есть, в неявном и очень упрощенном виде — в проверке света на каждой итерации цикла.

Давайте увидим это в 5-й группе тестов. Когда некоторые квартиры могут состоять из двух окон, решение, очевидно, меняют эти самые квартиры из двух окон. Тогда просто рассмотрим их отдельно. Обнаружить их можно при помощи аккуратного условия: если на текущей итерации окно i не последнее и при этом цвет окна с номером $i + 1$ совпадает с i-м, то на текущей итерации рассматривается левое окно квартиры с двумя окнами. В таком случае к переменной, считающей ответ, нужно прибавлять единицу, только если в обоих окнах горит свет. Нужно при этом пропустить следующую итерацию цикла, так как при стандартной реализации программа может посчитать, что она находится в очередной квартире с одним окном. В некоторых языках программирования для этого достаточно увеличить переменную цикла на 1. Там, где это сделать нельзя, можно написать аккуратное условие: если на текущей итерации окно i не первое и при этом цвет окна с номером $i-1$ совпадает с i-м, то на текущей итерации рассматривается правое окно квартиры с двумя окнами, а значит нужно эту итерацию пропустить.

Видно, что подсчёт числа окон с горящим светом спрятан в проверке света у квартир с двумя окнами. При этом такой подсчёт запускается на каждой итерации цикла, кроме итераций с пропуском, заново.

Как тогда можно изменить решение, чтобы получить полный балл? Условие заранее задать нельзя, так как длины квартир могут быть любыми. Но вспомним, что решение 2-й группы тестов работает для любой длины квартиры. Притом в решении 3-й группы тестов обнаружено, что разбиение, то есть точка, где заканчивается подсчёт, происходит при смене цвета окон.

Тогда давайте запускать подсчёт числа окон с включенным (или выключенным) светом по каж-

дой квартире явно — примерно так, как это делалось в решении второй группы. Для этого нужно написать вложенный цикл, который считает число окон с включенным (или с выключенным) светом до перехода в другую квартиру, то есть до смены цвета окон, либо до конца строки, если цикл был запущен в последней квартире. Такой цикл, очень похожий на решение 2-й группы тестов, заменит условные операторы, которые проверяли свет в окнах в решении 5-й группы тестов. Затем напишем условие, по которому будет обновляться ответ — оно либо остается таким же, как в решении 2-й группы, либо в нем меняется сравнение с числом n на сравнение с длиной квартиры (см. первый вариант полного решения). Остается только пропустить следующие окна уже рассмотренной квартиры, что можно сделать аналогично тому, как это делалось в 5-й группе тестов; проще всего оставить проверку цвета предыдущего окна — она работает для квартир любой длины.

В этой задаче $N \leqslant 10^5$, что ограничивало использование функций внутри цикла (чтобы не допустить квадратичного времени работы программы — когда на строки длины N требуется совершить число операций, сравнимое с N^2), и неявно подсказывало, что задача, скорее всего, может быть решена единственным циклом с какими-то условиями внутри. Однако в качестве второго варианта полного решения предлагается написать вложенный цикл, что обычно свидетельствует о квадратичном времени работы программы. Почему тогда это должно работать?

Спасает от квадратичного времени работы именно пропуск окон той квартиры, ответ для которой уже посчитан. Действительно, тогда вложенный цикл будет запускаться нечасто, причем важно то, что за все его запуски он обойдет все введенные символы только один раз, так как он считает включенный или выключенный свет по каждой квартире ровно один раз. Как следствие заметим, что возможный вердикт «`Time limit exceeded`» мог быть вызван именно отсутствием пропуска уже просмотренных квартир, перегоняя по приоритету ожидаемый вердикт «`Wrong answer`».

Критерии оценивания:

№	Баллы	Дополнительные ограничения	Необх. группы
1	12	$N = 1$	—
2	13	В доме ровно одна квартира	1
3	16	В доме не больше двух квартир	1 – 2
4	14	У каждой квартиры ровно одно окно	1
5	18	У каждой квартиры не более двух окон	1, 4
6	27	Без дополнительных ограничений	1 – 5

Баллы выставляются автоматически проверяющей системой.

Задача E. Произведение на отрезке

Попробуем сначала понять, что в целом происходит в задаче. Для этого можно решить первые две группы тестов, причем для их решения требуются только знания условных операторов. Действительно, все возможные тесты можно перебрать на листочке (их всего 12) и решить каждый из них как математическую задачу, например, полным перебором. Аналогично можно решить и третью группу тестов.

Однако как только становится ясным, что происходит в задаче и как написать полный перебор, следует реализовать его любым способом для прохождения групп 1 – 4. Например, функцию $s(x)$ можно реализовать только для однозначных и двузначных x как сумму двух чисел: остатка от деления x на 10 и целой части от деления x на 10. Эти вычисления не обязательно реализовывать отдельной функцией — можно просто считать это значение как формулу.

Решение начнем с перебора чисел l и r при помощи двух вложенных циклов, причем внутренний должен начать перебор числа r с $r = l$. Затем переберем все числа из отрезка $[l; r]$ еще одним вложенным циклом, и перемножим их между собой (для этого потребуется вспомогательная переменная; обратите внимание, что в отличие от суммы ее начальное значение равно 1, а не 0), а затем проверим делимость на 9 при помощи взятия остатка от деления на 9. Если произведение не поделилось, то к ответу нужно прибавить единицу за найденный отрезок. Так как результат не превосходит $s(1) \cdot s(2) \cdot \ldots \cdot s(12) = 2177280$, хранить вычисления можно в стандартном 32-битном целом числе.

Далее необходимо сделать несколько оптимизаций этого решения. Чтобы пройти группы 5–6 заметим, что текущий полный перебор делает много лишней работы. На итерации с числами (l, r), где $r > l$ число $s(l) \cdot s(l+1) \cdot \ldots \cdot s(r)$ будет посчитано циклом при помощи не менее, чем $r - l$ операций умножения. Однако на прошлой итерации (для чисел $(l, r-1)$) уже было посчитано число $s(l) \cdot s(l+1) \cdot \ldots \cdot s(r-1)$, и для получения требуемого его нужно только умножить на $s(r)$.

Пусть вспомогательная переменная будет сохранять свое значение между итерациями цикла, перебирающего r (для этого ее можно, например, инициализировать в теле внешнего цикла), тогда в его теле можно заменить цикл на единственную операцию — домножение вспомогательной переменной на $s(r)$, и вычисления будут корректными.

Остается только изменить способ, при помощи которого считается $s(x)$. Можно подобно тому, как эта функция реализовывалась ранее, написать версию для трёх- четырёхзначных x. Но можно и написать $s(x)$ для любых x. Для этого нужно завести дополнительную переменную, изначально равную 0, в которую просуммируем все цифры числа x при помощи цикла. Этот цикл должен идти до тех пор, пока x не обнулится. Обновление переменной в цикле происходит следующим образом: прибавим к ней последнюю цифру числа x при помощи взятия остатка от деления x на 10, а после этого уберем последнюю цифру числа x, заменив x на целую часть от деления x на 10. Такой алгоритм обнуляет x, что следует учитывать при реализации, однако его идея и реализация довольно простые.

Но важно отметить, что в 5–10 группах тестов произведение $s(l) \cdot s(l+1) \cdot \ldots \cdot s(r)$ может превышать даже 64-битный тип данных, притом время работы программы будет сильно увеличено, если использовать неограниченно длинные целые числа.

Отметим, что конечное решение не требует решения этой проблемы.

Более понятное решение проблемы с переполнением — хранить вместо произведения только вхождение степени тройки в произведение: пусть $s(l) \cdot s(l+1) \cdot \ldots \cdot s(r) = 3^k \cdot C$, тогда будем хранить только число k. Действительно, число делится на 9, если степень вхождения тройки в произведение не меньше 2 ($k \geqslant 2$). Но степень вхождения тройки в некоторое число $s(x)$ может быть нетрудно вычислено перебором при помощи цикла, а степень вхождения произведения считается как сумма степеней вхождения множителей.

Есть и другое решение. Вспомним следующий факт: если числа y и z дают одинаковый остаток от деления на любое натуральное число m, то при замене числа y на z в любом арифметическом выражении, операциями которого являются $+$, $-$ и/или $\cdot$, остаток от деления этого арифметического выражения на m не изменится.

Заметим, что числа z и $z \bmod 9$, где mod — операция взятия остатка от деления, дают одинаковый остаток при делении на 9, равный $z \bmod 9$. Значит, можно перемножать не числа $s(l), s(l+1), \ldots, s(r)$, а числа $(s(l) \bmod 9), (s(l+1) \bmod 9), \ldots, (s(r) \bmod 9)$. Более того, после каждого перемножения можно заменить промежуточный результат на остаток от деления этого промежуточного результата на 9. Но так как $y \bmod z < z$, во вспомогательной переменной можно всегда поддерживать результат, не превышающий $9 \cdot 9 = 81$. Таким образом можно решить проблему с переполнением.

Для решения 7–8 групп нужно придумать еще какую-нибудь оптимизацию. Тогда воспользуемся фактом о взятии остатка от деления немного с другой стороны. Для этого вспомним (обобщенный) признак делимости на 9, который утверждает, что $s(x)$ и x дают одинаковый остаток при делении на 9. Значит, посчитать остаток от деления $s(l) \cdot s(l+1) \cdot \ldots \cdot s(r)$ на 9 — это то же самое, что и посчитать остаток от деления $l \cdot (l+1) \cdot \ldots \cdot r$ на 9.

Но заметим, что среди любых 9 последовательных натуральных чисел обязательно встретится число, которое делится на 9. Но если такое число нашлось на отрезке $[l, r]$, то оно обязательно найдется на всех отрезках $[l, r']$ для всех $r' > r$. Тогда оптимизация состоит в следующем: не всегда нужно увеличивать правую границу r для заданной левой границы l до B — можно выйти из вложенного цикла еще в тот момент, когда произведение впервые поделилось на 9. Другой вариант: задать границей перебора r число $\min(l+8, B)$.

Чтобы решить задачу на полный балл, заметим, что все подходящие отрезки находятся между числами, кратными 9, и, более того, между любыми двумя соседними числами, кратными 9, которые содержатся в $[A; B]$, количество отрезков одинаково и равно 27.

Первая идея напрямую следует из того, что мы уже знаем про подходящие отрезки — если на отрезке есть число, кратное 9, то произведение чисел на этом отрезке точно делится на 9. Но тогда если разрезать весь отрезок $[A;B]$ числами кратными 9 (для определенности будем считать, что числа, делящиеся на 9, уничтожаются после разрезания), то для каждого отрезка из полученных задачу можно решать независимо, а итоговый ответ посчитать как сумму по всем отрезкам, которые получились разрезанием.

Вторая идея основана на том, что числа z и $z+9$ дают одинаковый остаток от деления на 9, а значит остаток от деления на 9 произведения чисел $l, l+1, \ldots, r$ равен остатку от деления на 9 произведения чисел $l+9, l+10, \ldots, r+9$. Можно показать, что ответ на задачу при $A=1, B=8$ равен 27, но тогда из рассуждений следует, что ответ при сдвиге границ на 9, то есть для $A=9k+1, B=9k+8$ при любом натуральном k, тоже равен 27.

Заметим, что тогда большинство отрезков, получившихся после разрезания, будут давать в общий ответ число 27 из-за того, что их границы были получены разрезаниями по числам, которые делятся на 9 (то есть их вид: $[9k+1; 9k+8]$). Поэтому остается только перебором посчитать ответ для самого левого и правого отрезка.

Понятно, что явно строить такое разбиение не нужно. Для решения задачи на полный балл нужно только выделить из $[A;B]$ самый длинный подотрезок $[L;R]$, начинающийся с числа, кратного 9, и заканчивающийся числом, кратным 9. Тогда результат на $[L;R]$ можно посчитать как результат на отрезке $[1;8]$ (равный 27), умноженный на $(R-L)/9$. Оставшиеся подотрезки с начала и с конца можно проверить простым перебором, например как в решении 4-й подзадачи.

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		A	B	
1	8	$A=1$	$B \leqslant 5$	—
2	11	$A=1$	$B \leqslant 12$	1
3	4	$A \leqslant B$	$B \leqslant 5$	1
4	18	$A \leqslant B$	$B \leqslant 12$	1–3
5	15	$A \leqslant B$	$B \leqslant 500$	1–4
6	5	$A \leqslant B$	$B \leqslant 2\,000$	1–5
7	6	$A \leqslant B$	$B \leqslant 50\,000$	1–6
8	7	$A \leqslant B$	$B \leqslant 150\,000$	1–7
9	14	$A=1$	$B \leqslant 5 \cdot 10^8$	1, 2
10	12	$A \leqslant B$	$B \leqslant 5 \cdot 10^8$	1–9

Баллы выставляются автоматически проверяющей системой