Задача A. Три сына

Для решения этой задачи требуется считать целые числа A, B и C, а затем вывести $A + B + C$.

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 8 — ответа на задачу при $A = 8, B = 0, C = 0$.

Во 2-й группе тестов $B = 0$ и $C = 0$, поэтому достаточно считать только первое число и вывести ответ, равный A.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		A	B	C	
1	12	$A = 8$	$B = 0$	$C = 0$	—
2	25	$A \leqslant 100$	$B = 0$	$C = 0$	1
3	63	$A \leqslant 100$	$B \leqslant 100$	$C \leqslant 100$	1, 2

Баллы выставляются автоматически проверяющей системой.

Задача B. Голландские вафли

Если $t = a + b$ и $k = p$, то Ваня может сделать не больше 1 вафли, а 1 вафлю можно сделать всегда. Поэтому чтобы решить 1-ю группу, достаточно вывести единицу.

Во 2-й группе также нельзя испечь больше 1 вафли, так как $k = p$. Поэтому задача состоит в том, чтобы проверить хватит ли нам времени, чтобы испечь 1 вафлю, то есть проверить, что $t \geqslant a + b$.

3-я группа решается аналогично 2-й, только здесь надо проверить, хватит ли теста, чтобы испечь 1 вафлю, то есть что $p \geqslant k$.

В 4 группе теста хватит максимум на 2 вафли. Давайте посмотрим, хватит ли Ване времени на приготовление одной ($t \geqslant a + b$) или двух ($t \geqslant 2 \cdot (a + b)$) вафель. Разобрав несколько случаев, получаем ответ.

Чтобы сдать эту задачу на полный балл, давайте посчитаем, на сколько вафлей хватает времени и на сколько вафлей хватает теста: $c_1 = t//(a + b)$, $c_2 = k//p$, где // обозначает целочисленное деление. Тогда ответом будет $\min(c_1, c_2)$. Итоговая сложность решения — $\mathcal{O}(1)$.

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		t	k	
1	9	$t = a + b$	$k = p$	—
2	14	$t \leqslant 2 \cdot 10^9$	$k = p$	1
3	14	$t = a + b$	$k \leqslant 2 \cdot 10^9$	1
4	21	$t \leqslant 2 \cdot 10^9$	$k \leqslant 2p$	1, 2
5	42	$t \leqslant 2 \cdot 10^9$	$k \leqslant 2 \cdot 10^9$	1-4

Баллы выставляются автоматически проверяющей системой.

Задача C. Вася и цветочки

В 1-й группе при заданных ограничениях Вася может только проходить каждое поле друг за другом раз в день. Значит, в i-й день он будет получать единицу недовольства, если посетит поле, которое ему нравится — такой уж странный Вася. Так, за каждую единицу в строке s Вася получит одну единицу недовольства, поэтому, чтобы посчитать ответ, посчитаем при помощи цикла или встроенной в стандартную библиотеку языка функции число символов «1» в строке s.

Во 2-й группе выгоднее всего в первый же день пройти весь маршрут. Тогда недовольство Васи будет равно 1, если в строке было нечетное число единиц, и 0 иначе. Данное решение отличается от решения первой группы тестов только выводом.

Поймем, почему это оптимальная стратегия. Если Вася получит нулевое недовольство, то меньшее недовольство точно получить нельзя. Но если эта стратегия даст нам единицу недовольства, почему нет маршрута, который не принесет ни единицы недовольства?

Если этот алгоритм дает единицу в качестве ответа, то всего в строке s нечетное число единиц. Чтобы получить в результате ноль единиц недовольства, Васе нужно организовать маршрут так, чтобы каждый день проходить четное число полей, которые ему нравятся. Но придерживаясь такой стратегии он суммарно пройдет тоже четное число полей, а должен пройти нечетное! Поэтому предложенная стратегия оптимальна.

В 3-й группе можно применить следующий алгоритм: если два следующих поля нравятся Васе, то в этот день посетим 2 поля, иначе только 1.

Обоснуем эту стратегию. Чтобы минимизировать суммарное недовольство, нужно как можно меньше дней проходить нечетное число полей, которые нравятся Васе. В этой группе тестов это значит, что если Вася встретит поле, которое ему нравится, то единственный случай, когда он не получит единицу недовольства — если за ним снова поле, которое ему нравится, и он пройдет за один день оба поля. Отметим также, что в остальных случаях проходить два поля за один день бессмысленно: ответ это улучшить не может, и Васе безразлично, за сколько дней он прошел весь маршрут. Зато это заметно упрощает реализацию.

Чтобы проэмулировать эту стратегию, понадобится переменная, в которой будем считать общее недовольство Васи, и цикл, на i-м шаге которого будем обрабатывать ситуацию: «Вася прошел все поля до i-го, то есть следующее поле, которое он должен посетить — i-е». Посмотрим на текущее поле и на следующее, и если хотя бы один символ $s[i]$ или $s[i+1]$ равен `0`, то следует посетить только i-е поле, то есть перейти к следующему шагу цикла (например, при помощи оператора `continue`), перед этим, если нужно, добавив единицу недовольства к общему недовольству. Если же $s[i] = s[i+1] = $ `1`, то нужно пропустить следующий шаг цикла, и сразу продолжить с шага $i + 2$.

Чтобы такое реализовать, заведем вспомогательную переменную `skip`, которая должна показывать, сколько шагов цикла следует пропустить. Будем начинать цикл с того, что проверим, не нужно ли текущий шаг пропустить: если `skip` > 0, уменьшим `skip` на единицу и начнем следующий шаг (`continue`) — этот шаг цикла пропущен. Если же `skip` $= 0$, то шаг пропускать не нужно. Тогда при встрече двух единиц подряд, чтобы пропустить следующий шаг цикла, достаточно присвоить `skip` значение 1.

Отметим также, что проверку *следующего* поля нужно делать аккуратно: если текущий шаг цикла последний, то i — последнее поле, за ним не стоит следующего. Если этот случай не учесть и всегда обращаться к $s[i + 1]$, то программа может аварийно завершиться.

Немного улучшив решение, можно придумать оптимальную стратегию и решить задачу на полный балл. Если Вася способен ходить на k полей за один день, то единственный случай, когда это однозначно лучше, чем ходить по одному полю за день — это ситуация, когда следующее поле (которое Вася еще не посетил) ему нравится, и нужно посетить четное число полей, чтобы не увеличивать Васино недовольство. Тогда нет смысла проходить четное число полей, большее 2, ведь один день, за который Вася проходит 4, 6, 8, и т д. полей можно разбить на 2, 3, 4, и т д. дня, в которые он проходит по два поля. Это упрощает реализацию.

Тогда стратегия Васи в следующем: он посещает по одному полю в день до тех пор, пока не встретит поле, которое ему нравится. Когда же Вася встречает поле, которое ему нравится, проверим, сможет ли он посетить следующее за ним поле, которое ему тоже нравится. Если да, посетим за текущий день все поля между этими двумя, а если нет — посетим только одно поле.

Возможность пропускать любое число шагов уже реализовано при помощи переменной `skip`. А для поиска следующей `1` в строке s можно запустить вложенный цикл от позиции $i+1$ до $i+k$. Это правильный подход, но неэффективный: сложность такого решения составит $\mathcal{O}(nk)$.

Рабочий способ ускорить поиск следующей единицы — заранее подсчитать в массиве h позицию следующей единицы для каждого индекса i. Чтобы это сделать, запустим цикл от конца строки к началу и будем поддерживать во вспомогательной переменной `last` последнюю позицию, где встретилась единица. Тогда на каждом шаге цикла i сначала присвоим $h[i]$ значение `last`, а потом, если $s[i] = $ `1`, запишем в `last` значение i.

Теперь исходный алгоритм состоит только в том, чтобы на i-м шаге, если $s[i] = $ `1`, посмотреть

на значение $h[i]$ и убедиться, что расстояние от позиции i до позиции $h[i]$ не больше, чем k — тогда Вася может за текущий день посетить два поля, которые ему нравятся, что ему и следует сделать. Сложность такого решения: $\mathcal{O}(n)$.

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		n	k	
1	7	$n \leqslant 1000$	$k = 1$	—
2	9	$n \leqslant 5 \cdot 10^5$	$k = n$	—
3	28	$n \leqslant 5 \cdot 10^5$	$k \leqslant 2$	1
4	34	$n \leqslant 1000$	$k \leqslant 1000$	1
5	22	$n \leqslant 5 \cdot 10^5$	$k \leqslant 5 \cdot 10^5$	1–4

Баллы выставляются автоматически проверяющей системой.

Задача D. Урок информатики

В 1-й группе можно просто перебрать числа a, b, c, d от 0 до M при помощи вложенных циклов и проверить для каждой перебираемой четверки чисел, что $\frac{a+b}{c+d} = \frac{e}{f}$. Сложность решения: $\mathcal{O}(M^4)$.

Во 2-й группе $e = 0$, поэтому $a+b = 0$, а значит, $a = 0$ и $b = 0$. Тогда давайте переберем числа c и d и посчитаем количество пар, подходящих по условию. Так как числители уже таковы, что условие всегда будет при подстановке превращаться в «$0 = 0$», то подходит каждая пара чисел (c, d), при которой условие имеет смысл. Число пар можно посчитать при помощи комбинаторных формул: их будет ровно $(M+1)^2 - 1$. Действительно, каждому числу из пары доступен $M+1$ вариант значения (любое целое от 0 до M), поэтому основная формула $(M+1) \cdot (M+1)$, из которой нужно вычесть единицу, так как не подходит только одна пара: $c = 0$ и $d = 0$, потому что при этих значениях знаменатель дроби равен нулю.

Чтобы решить 3-ю группу, будем перебирать только числа a и b. Тогда из $\frac{a+b}{c+d} = \frac{e}{f}$ следует $c + d = \frac{(a+b) \cdot f}{e}$. При переборе чисел a и b значение $\frac{(a+b) \cdot f}{e}$ можно посчитать, пусть оно равно k.

Далее нужно посчитать число пар (c, d) таких, что $c+d = k$ при помощи комбинаторных формул. Если $0 < k \leqslant M$, то подходит $k+1$ вариант: $\{c = 0, d = k\}$, $\{c = 1, d = k-1\}$, …, $\{c = k, d = 0\}$. Если $k > M$, то подходит $2M - k + 1$ пар: $\{c = M, d = k - M\}$, $\{c = M-1, d = k - M + 1\}$, …, $\{c = k - M, d = M\}$. Сложность этого решения: $\mathcal{O}(M^2)$.

В 4-й группе переберем положительное k, на которое следует домножить дробь $\frac{e}{f}$, чтобы получить $\frac{a+b}{c+d}$, то есть такое k, что $a + b = k \cdot e$, $c + d = k \cdot f$.

Давайте докажем, что k в этой группе тестов должно быть целым числом. Конечно, k — рациональное, так как иначе число $a + b = e \cdot k$ было бы иррациональным, а нам нужно, чтобы $a + b$ было рациональным, и более того — целым. Тогда докажем от противного: пусть k такое *нецелое* число, что $a+b = k \cdot e$ и $c+d = k \cdot f$. Так как k — рациональное, то представим k в виде несократимой дроби: $k = \frac{p}{q}$. Значит, $\frac{p \cdot e}{q}$ и $\frac{p \cdot f}{q}$ равны $a + b$ и $c + d$ соответственно. Но тогда e и f делятся на q, потому что p и q взаимно просты, и $q \neq 1$, так как мы взяли такое k, что оно нецелое. Следовательно дробь $\frac{e}{f}$ можно сократить на q, но $\frac{e}{f}$ — несократимая дробь по условию группы. Противоречие. Получается, что k — целое положительное число.

Тогда k можно перебирать при помощи цикла. Для каждого k задача разбивается на две: нужно найти число пар (a, b), что $a + b = k \cdot e$ и найти число пар (c, d), что $c + d = k \cdot f$. Но эти задачи мы уже умеем решать: см. решение 3-й группы.

В 5-й группе до того, как перебирать, нужно сократить дробь $\frac{e}{f}$. Сделать это можно с помощью нахождения наибольшего общего делителя алгоритмом Евклида, после чего задача сводится к шестой группе тестов. Итоговая сложность: $\mathcal{O}(M + \log min(e, f)))$.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		e	M	дополнительно	
1	16	$e \leqslant 10^5$	$M \leqslant 50$	$\frac{e}{f}$ — несократимая дробь	—
2	16	$e = 0$	$M \leqslant 5 \cdot 10^5$	—	—
3	31	$e \leqslant 10^5$	$M \leqslant 1000$	$\frac{e}{f}$ — несократимая дробь	1
4	27	$e \leqslant 10^5$	$M \leqslant 5 \cdot 10^5$	$\frac{e}{f}$ — несократимая дробь	1, 3
5	10	$e \leqslant 10^5$	$M \leqslant 5 \cdot 10^5$	—	1–4

Баллы выставляются автоматически проверяющей системой.

Задача E. Странная система

1 группа. В данной группе количество членов жюри равно количеству карточек с оценками. Следовательно, жюри может взять только все карточки для оценки. Поэтому надо просто посчитать модуль разности между новой системой оценки и старой по всем карточкам, затем его вывести.

Чтобы найти балл по старой системе оценивания, нужно найти $a_1 + a_2 + a_3$ и поделить на 3. Сумму посчитать нужно, а результат деления в этой задаче считать не надо, так как ответ нужно вывести в качестве дроби. Запомним сумму в переменной `old`.

Чтобы найти балл по новой системе оценивания, воспользуемся найденной суммой $a_1+a_2+a_3$. Из нее нужно вычесть максимум и минимум среди a_1, a_2, a_3. Найти максимум и минимум в этой группе тестов можно при помощи условных операторов: сравним a_1 с a_2, а затем меньший или больший из этих элементов сравним с a_3. После вычитания максимума и минимума останется результат по новой системе оценивания, который запомним в переменную `new`.

Как считать ответ. Эта идея будет использована во всех группах тестов. Всегда результат по старой системе оценивания и по новой можно представить как две дроби: $\frac{old}{k}$ и $\frac{new}{k-2}$, где old и new — суммы чисел на каких-то k или $k-2$ карточках. Абсолютная разность этих двух чисел равна $\left|\frac{old}{k} - \frac{new}{k-2}\right| = \frac{|old \cdot (k-2) - new \cdot k|}{k \cdot (k-2)}$. Ответ всегда можно вывести ровно в таком формате, без того, чтобы делать дробь несократимой.

Так, в первой группе тестов далее нужно вывести $|old - 3 \cdot new|$ (модуль во многих языках программирования реализован во встроенной библиотеке как функция `abs()`), затем без пробелов вывести слеш (`/`), а за ним число 3. Итоговая сложность решения: $\mathcal{O}(1)$

2 группа. Так как количество жюри равно трем, можно перебрать, какие оценки возьмет каждый из жюри при помощи вложенных циклов. Для перебираемой тройки индексов карточек (i_1, i_2, i_3) проверим, что индексы попарно различны, и после этого узнаем, какая будет разница между результатами по новой системе и по старой так, как это делалось при решении первой группы тестов. Заведем вспомогательную переменную, где будем хранить наибольшее отличие, и для каждой подходящей тройки обновим значение в этой переменной как максимум из текущего значения в ней и результата при выборе карточек $a_{i_1}, a_{i_2}, a_{i_3}$.

Отметим, что под «результатом» удобно понимать не дробь и не вещественное число, а число $|old \cdot (k-2) - new \cdot k|$. Действительно, так как при любом выборе карточек итоговое значение будет выводиться в виде дроби с фиксированным знаменателем, равным $k \cdot (k-2)$, в действительности нужно минимизировать числитель дроби, который можно представить в таком виде. Итоговая сложность решения: $\mathcal{O}(n^k)$

3 группа. Распишем, чему равны old и new в общем случае при некоторых выбранных карточках. Пусть sum — сумма всех оценок жюри, mn и mx — минимум и максимум среди этих оценок. $\left|\frac{old}{k} - \frac{new}{k-2}\right| = \left|\frac{sum}{k} - \frac{sum - mn - mx}{k-2}\right| = \left|\frac{sum \cdot (k-2) - sum \cdot k + k \cdot mn + k \cdot mx}{k(k-2)}\right| = \left|\frac{k \cdot mn + k \cdot mx - 2sum}{k(k-2)}\right| = \left|\frac{(k-2)(mn+mx) - 2sum'}{k(k-2)}\right|$, где $sum' = sum - mn - mx$, то есть сумма оценок без максимума и минимума. Это значит, что результат зависит только значений mn, mx и sum'.

Так как в итоговом выражении фигурирует модуль, наибольшее значение это выражение может принимать при тех значениях mn, mx и sum', при которых выражение $(k-2)(mn+mx) - 2sum'$ принимает как наибольшее, так и наименьшее значение.

Назначим двух членов жюри «главными»: пусть карточка первого будет самой большой, а карточка второго — самой маленькой. Переберем все возможные варианты, какие карточки они могут

взять, и для каждого из них найдем максимальный результат. При таком переборе значения mn и mx определяются автоматически как значения на карточках, которые взяли члены жюри. Теперь нужно понять, как за разумное время искать sum'.

Чтобы минимизировать/максимизировать абсолютную разность между баллами по новой и старой системам оценивания, нужно максимизировать/минимизировать значение sum'. Для начала оставим только те карточки, на которых числа a_i не меньше mn и не больше mx — назовем эти числа *подходящими*. Если таких меньше $k-2$, то выбор карточек для «главных» членов жюри неудачен, нужно продолжить перебор.

Сначала посчитаем результат, если sum' решено минимизировать. Тогда sum' следует посчитать, как сумму $k-2$ самых маленьких значений среди подходящих. Потом посчитаем результат, если sum' решено максимизировать, взяв $k-2$ самых больших значения среди подходящих. Но как найти эти числа?

Для этого отсортируем массив $a_1, a_2, \ldots, a_n$. Тогда и перебор карточек «главных» членов жюри проще: достаточно, чтобы индекс карточки первого (после сортировки) был хотя бы на $k-1$ больше, чем индекс карточки второго. Пусть первому досталась карточка $i_{\max}$, а второму — $i_{\min}$. Тогда понятно, что $k-2$ самых маленьких значения среди подходящих — это $k-2$ числа, стоящие после $i_{\min}$-го в отсортированном массиве $a_1, a_2, \ldots, a_n$, то есть числа $a_{i_{\min}+1}, a_{i_{\min}+2}, \ldots$ Аналогично, $k-2$ самых больших значения среди подходящих — это $k-2$ числа, стоящие до $i_{\max}$-го. Посчитать искомые суммы можно при помощи вложенного цикла.

Теперь при таком переборе для каждой пары карточек у «главных» членов жюри известно два значения, максимум из которых и есть наибольший результат, а ответом будет максимум по всем результатам. Напомним, что результатом мы называем $|old \cdot (k-2) - new \cdot k| = |(k-2)(mn+mx) - 2sum'|$. Сложность решения: $\mathcal{O}(n^2 k)$.

4 группа. Улучшим решение предыдущей подгруппы. Заметим, что всегда требуется узнать сумму *последовательных* чисел, что позволяет ускорить работу программы, воспользовавшись префиксными суммами — посчитаем во вспомогательный массив pf суммы на префиксах массива: $pf[i] = a_1 + a_2 + \ldots + a_i$, и тогда сумму $a_{l+1} + a_{l+2} + \ldots + a_r$ можно посчитать как $pf[r] - pf[l]$. Так, сложность решения: $\mathcal{O}(n^2)$, что для некоторых языков достаточно, чтобы пройти эту группу тестов.

5 группа. Если $a_1 = a_2 = \ldots = a_n$, то ответ будет «0/1». Теперь рассмотрим случай, когда есть две карточки с различными оценками. Пусть оценки будут x и y, где $x < y$. Тогда обозначим за cnt_x и cnt_y, сколько члены жюри в итоговую оценку взяли карточек со значениями x и y соответственно.

Распишем разницу итоговых оценок. Если $\min(cnt_x, cnt_y) \geqslant 1$, то она равна $\frac{old \cdot (k-2) - new \cdot k}{k \cdot (k-2)} = \frac{(k-2)(x \cdot cnt_x + y \cdot cnt_y) - k \cdot (x \cdot cnt_x + y \cdot cnt_y)}{k(k-2)} = \frac{(k-2)(x \cdot (k - cnt_y) + y \cdot cnt_y) - k \cdot (x \cdot (k - cnt_y) + y \cdot cnt_y)}{k(k-2)}$. Заметим, что итоговая разница зависит только от cnt_y. Надо посчитать минимум и максимум по значениям этого выражения. Для этого можно просто перебрать все возможные значения cnt_y из промежутка от $\min(1; k - CNT_x)$ по $\max(k-1; CNT_y)$, где CNT_x и CNT_y — количество оценок x и y на карточках. Ответом на задачу будет максимальное по модулю значение выражения выше. Сложность такого решения: $\mathcal{O}(n)$.

6 группа. Продолжим улучшать решение 4 группы. Заметим, что при подсчете максимума выражения, всегда выгодно брать $mn = \min(a_i) = a_1$ (после сортировки), а если считаем минимум, то надо брать $mx = \max(a_i) = a_n$ (после сортировки). Тогда остается только перебрать оценку оставшегося из «главных» членов жюри и посчитать sum' при помощи префиксных сумм.

Итоговая сложность: $\mathcal{O}(n + \texttt{сложность сортировки})$, где сложность сортировки зависит от способа, при помощи которого массив $a_1, a_2, \ldots, a_n$ сортируется. Для решения задачи на полный балл нужно, чтобы сортировка была достаточно быстрой — для этого подойдет, например, сортировка слиянием или быстрая сортировка, а также в некоторых языках подходит функция сортировки из стандартной библиотеки.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		n	k	доп. ограничения	
1	8	$n = 3$	$k = 3$	—	—
2	15	$n \leqslant 50$	$k = 3$	—	1
3	29	$n \leqslant 100$	$k \leqslant n$	—	1, 2
4	5	$n \leqslant 5000$	$k \leqslant n$	—	1, 2, 3
5	20	$n \leqslant 10^5$	$k \leqslant n$	$\lvert\{a_1, a_2 \dots a_n\}\rvert \leqslant 2$	—
6	23	$n \leqslant 10^5$	$k \leqslant n$	—	1–5

Баллы выставляются автоматически проверяющей системой.