

Муниципальный этап
Всероссийской олимпиады школьников
по информатике

в 2015 – 2016 учебном году

Разборы решений и идеи тестов

Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2015 – 2016 учебном году
8 класс

Время выполнения задач — 4 часа

Ограничение по времени — 2 секунды на тест

Ограничение по памяти — 64 мегабайта

8.1. «Большее произведение». Вводятся три целых числа a , b , c . Вывести большее из произведений $a \cdot b$ и $b \cdot c$.

Формат входа: В единственной строке через пробел заданы три целых числа a , b , c , по модулю не превосходящие 30000.

Формат выхода: Выведите единственное целое число — наибольшее из указанных произведений.

Пример

Вход: Выход:

5 2 4 10

8.2. «Суммарная разность». На уроках информатики Петя Торопыжкин начал изучать массивы. Он придумал следующую операцию с массивом: находится разность второго и первого элементов массива, затем третьего и второго, затем четвёртого и третьего, и т.д. до разности последнего и предпоследнего элементов. После все найденные разности суммируются. Помогите Пете, написав программу, которая проделывает указанную операцию над заданным массивом.

Формат входа: В первой строке задано целое число n — количество элементов в массиве ($2 \leq n \leq 1000$). В следующей строке через пробел задано n целых чисел, каждое по модулю не превосходит 10^6 .

Формат выхода: Выведите единственное целое число — сумму попарных разностей соседних элементов массива.

Пример

Вход: Выход:

4 -4

5 2 6 1

Примечание: Результат получается следующим образом: $2 - 5 = -3$, $6 - 2 = 4$, $1 - 6 = -5$, $(-3) + 4 + (-5) = -4$.

8.3. «Наибольший остаток». Часто на уроках информатики Петя Торопыжкин придумывает разные операции для сравнения натуральных чисел. Вот и теперь он предложил новую операцию: одно число больше другого, если его остаток от деления на 2015 больше остатка от деления на 2015 второго числа. Напишите

программу, которая ищет максимальный в смысле Петинского сравнения элемент в массиве чисел. Если таких чисел несколько, укажите любое из них.

Формат входа: В первой строке задано целое число n — количество элементов в массиве ($2 \leq n \leq 1000$). В следующей строке через пробел задано n неотрицательных целых чисел, каждое из которых не превосходит 10^8 .

Формат выхода: Выведите какое-нибудь число из заданного набора, дающее максимальный остаток при делении на 2015.

Пример

Вход: Выход:

3 2016

1 0 2016

8.4. «Сколько символов». На парте в кабинете английского языка Петя Торопыжкин обнаружил длинное слово (состоящее только из букв латиницы — кабинет-то английского языка!). Петин сосед по парте выбрал одну букву из латинского алфавита и спросил Петю, сколько раз эта буква встречается в обнаруженном слове. Помогите Пете посчитать количество вхождений, напишите соответствующую программу.

Формат входа: В первой строке задан единственный символ. Во второй строке задана строка. Все символы — заглавные латинские буквы. Длина строки не превосходит 255 символов.

Формат выхода: Выведите единственное неотрицательное целое число — количество вхождений символа в строку.

Пример

Вход: Выход:

A 4

ABCASVAA

8.5. «Крайние элементы». Петя Торопыжкин пошёл в поход. Чтобы поставить палатку, ему нужны три длинных палки (чтобы поставить распорки) и три коротких (для колышков). На месте стоянки нашлось n палок. Пете нужно выбрать три самые длинные из них и три самые короткие. Помогите ему, напишите соответствующую программу.

Формат входа: В первой строке задано целое число n — количество найденных палок ($6 \leq n \leq 1000$). В следующей строке через пробел задано n целых чисел — длины найденных палок (в каком-то порядке). Каждая длина есть натуральное число, не превосходящее 10^8 .

Формат выхода: Выведите в порядке возрастания шесть чисел, разделённых пробелами: три наименьших и три наибольших длины палок среди имеющегося набора.

Пример

Вход:

7

6 1 5 3 2 4 7

Выход:

1 2 3 5 6 7

Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2015–2016 учебном году
8 класс. Разбор решений и идеи тестов

8.1. «Большее произведение». Вводятся три целых числа a , b , c . Вывести большее из произведений $a \cdot b$ и $b \cdot c$.

Данная задача носит утешительный характер и подразумевает прямое написание требуемого небольшого алгоритма.

Также возможен путь, основанный на сравнении величин a и c , то есть на проверке неравенства $a > c$, полученного из неравенства $ab > bc$ сокращением b (если $b \neq 0$). Однако здесь надо принять во внимание знак величины b .

Идеи тестов:

- 1–5. Случайные тесты, в которых произведение входит в тип `short int`. Есть тест с $b = 0$.
- 6–10. Случайные тесты, в которых произведение входит в тип `int`. Есть тест с $b = 0$.

8.2. «Суммарная разность». На уроках информатики Петя Торопыжский начал изучать массивы. Он придумал следующую операцию с массивом: находится разность второго и первого элементов массива, затем третьего и второго, затем четвёртого и третьего, и т.д. до разности последнего и предпоследнего элементов. После все найденные разности суммируются. Помогите Пете, написав программу, которая проделывает указанную операцию над заданным массивом.

Данная задача представляет два различных пути решения, один из которых весьма прост в реализации. Прямолинейное решение подразумевает циклическое считывание элементов массива (с запоминанием предыдущего элемента), нахождение и суммирование разностей.

Однако, если задуматься о сути проделываемой операции, то можно понять, что искомый результат есть разность последнего и первого элементов набора. Действительно, искомая величина есть

$$\begin{aligned} S &= (a_2 - a_1) + (a_3 - a_2) + (a_4 - a_3) + \dots + (a_{n-1} - a_{n-2}) + (a_n - a_{n-1}) = \\ &= a_2 - a_1 + a_3 - a_2 + a_4 - a_3 + \dots + a_{n-1} - a_{n-2} + a_n - a_{n-1} = a_n - a_1. \end{aligned}$$

Последнее равенство получаем потому, что в выражении в последней строке все слагаемые, кроме a_1 и a_n , встречаются дважды с разными знаками и взаимно уничтожаются.

Идеи тестов:

- 1–5. Случайные тесты, где величины a_i по модулю не превосходят 1000 и все промежуточные результаты при прямолинейном вычислении гарантированно входят в тип `short int`.
- 6–10. Случайные тесты, где величины a_i большие и промежуточные вычисления могут не войти в тип `short int`.

8.3. «Наибольший остаток». Часто на уроках информатики Петя Торопыжкин придумывает разные операции для сравнения натуральных чисел. Вот и теперь он предложил новую операцию: одно число больше другого, если его остаток от деления на 2015 больше остатка от деления на 2015 второго числа. Напишите программу, которая ищет максимальный в смысле Петинского сравнения элемент в массиве чисел. Если таких чисел несколько, укажите любое из них.

С точки зрения программного комитета, задача имеет средний уровень сложности для восьмиклассников. В основе решения лежит базовый алгоритм поиска максимального числа в наборе, несколько усложнённый тем, что нужно применять нестандартное сравнение чисел.

Идеи тестов:

- 1–5. Случайные тесты, максимум единственен.
- 6–10. Случайные тесты, максимум неединственен.

8.4. «Сколько символов». На парте в кабинете английского языка Петя Торопыжкин обнаружил длинное слово (состоящее только из букв латиницы — кабинет-то английского языка!). Петин сосед по парте выбрал одну букву из латинского алфавита и спросил Петю, сколько раз эта буква встречается в обнаруженном слове. Помогите Пете посчитать количество вхождений, напишите соответствующую программу.

Сложность задачи составляет лишь работа со строками, которая непривычная для восьмиклассников. Решение прямолинейно: считываем символ, строку и, проходя по символам строки, подсчитываем количество вхождений данного символа. Строка может быть пустой — это не запрещено условием.

Идеи тестов:

1. Минимальный тест: пустая строка.
2. Минимальный тест: строка длины 1, символ в строку не входит.
3. Минимальный тест: строка длины 1, символ в строку входит.
4. Максимальный тест: строка длины 255, не все символы одинаковы, символ в строку не входит.

5. Максимальный тест: строка длины 255, не все символы одинаковы, символ в строку входит.
6. Максимальный тест: строка длины 255, все символы одинаковые, символ в строку не входит.
7. Максимальный тест: строка длины 255, все символы одинаковые, символ в строку входит.
- 8–20. Случайные тесты.

8.5. «Крайние элементы». *Петя Торопыжкин пошёл в поход. Чтобы поставить палатку, ему нужны три длинных палки (чтобы поставить распорки) и три коротких (для колышков). На месте стоянки нашлось n палок. Пете нужно выбрать три самые длинные из них и три самые короткие. Помогите ему, напишите соответствующую программу.*

По мнению программного комитета. данная задача для восьмиклассников является сложной.

Разумными являются два пути решения. Первый — отсортировать массив и выдать первые три и последние три элемента. Второй — написать модифицированный алгоритм поиска минимального и максимального элементов, который будет вычислять не один, а три экстремальных элемента.

Идеи тестов:

1. Минимальный тест: $n = 6$. Все числа различны.
2. Минимальный тест: $n = 6$. Минимальные числа совпадают между собой и максимальные числа совпадают между собой.
3. Минимальный тест: $n = 6$. Все числа совпадают.
- 4–6. Максимальные тесты: $n = 1000$, то же, что в тестах 1–3.
- 7–25. Случайные тесты. Присутствуют тесты, в которых требуемые элементы различны и одинаковы.

Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2015 – 2016 учебном году
9 класс

Время выполнения задач — 4 часа

Ограничение по времени — 2 секунды на тест

Ограничение по памяти — 64 мегабайта

9.1. «Наибольший хвост». Заданы три натуральных числа n_1, n_2, n_3 , не меньших 10. Выведите то из них, две последние цифры которого, будучи рассмотрены в том же порядке, что и в десятичной записи числа, дают наибольшее двузначное число.

Формат входа: В единственной строке через пробел заданы три натуральных числа n_1, n_2, n_3 ($10 \leq n_i \leq 10^6$).

Формат выхода: Выведите единственное число, имеющее наибольшее двузначное число в разряде десятков и единиц. Если два или все три числа имеют одинаковые числа в этих разрядах, выведите любое из них.

Пример

<u>Вход:</u>	<u>Выход:</u>
198 691 19 198	

9.2. «Победа в бою». В свободное от уроков и тренировок по плаванию время Петя Торопыжкин играет в компьютерные игры. В очередном бою у Пети имеется отряд из k_1 пехотинцев, каждый из которых имеет силу удара s_1 и здоровье h_1 . У компьютерного противника также имеется отряд из k_2 пехотинцев, каждый с ударом s_2 и здоровьем h_2 . Отряд Пети первым наносит удар по отряду противника, затем атакует отряд противника, потом снова Петин отряд и т.д. Отряд из k человек (неважно, есть ли среди них раненый боец или нет) с ударом s наносит $k \cdot s$ единиц урона, которые последовательно вычитаются из здоровья наиболее раненого члена атакуемого отряда. Если единиц урона больше, чем здоровье такого члена отряда, то он умирает и здоровье начинает списываться у другого солдата (который также может погибнуть, если урона слишком осталось много) — до тех пор, пока есть живые солдаты или есть очки урона.

Напишите программу, которая определяет по указанным входным данным, какой отряд выживет в таком бою и сколько солдат останется в его составе.

Формат входа: В первой входной строке через пробел указаны три целых числа k_1, s_1, h_1 . Во второй строке так же через пробел указаны числа k_2, s_2, h_2 . При этом $1 \leq k_i, s_i, h_i \leq 1000$.

Формат выхода: Выведите через пробел два целых числа. Первое — 1, если выживет Петин отряд, или 2, если выиграет противник. Второе число — численность выжившего отряда (включая и раненого солдата, если такой есть).

Пример 1		Пример 2	
<u>Вход:</u>	<u>Выход:</u>	<u>Вход:</u>	<u>Выход:</u>
1 1 100	1 1	1 1 100	2 1
1 1 100		1 1 101	

9.3. «Новый химический элемент». Сидя на уроке химии, Петя Торопыжкин мечтал о том, как он откроет новый химический элемент. Потом он задумался о его названии и обозначении. И тут он заметил на парте в кабинете химии длинное слово, состоящее из заглавных латинских букв, и решил, что обозначением будет сочетание двух последовательных букв из увиденного слова. Но сочетаний много, и Петя решил выбрать из них минимальное в лексикографическом порядке.

Напомним, что лексикографический порядок сравнения слов — это порядок, принятый в словарях: слова сравниваются по первым буквам, при их равенстве по вторым, при равенстве вторых — по третьим и т.д., пока будет найдена несовпадающая пара букв, которая определит порядок слов, или одно из слов не кончится; в последнем случае более короткое слово, совпадающее с началом более длинного, считается меньшим.

Помогите Пете, написав программу, которая по введённому слову найдёт минимальное в лексикографическом порядке двухбуквенное подслово.

Формат входа: В единственной строке задано слово, состоящее из заглавных букв латиницы. Длина слова не менее 2 и не более 255 символов.

Формат выхода: Выведите единственную строку, содержащую минимальное в лексикографическом порядке двухбуквенное подслово входной строки.

Пример

<u>Вход:</u>	<u>Выход:</u>
ZCYBWA	BW

9.4. «Странное сравнение». На математическом кружке Петя Торопыжкин прослушал лекцию на тему отношений порядка и узнал, что целые числа можно сравнивать не только так, как это принято на уроках математики. Например, можно считать, что любое чётное число меньше любого нечётного, а пару чётных чисел или пару нечётных чисел уже сравнивать по обычным правилам — и это тоже будет порядком на натуральных числах! Правда, непривычным. В портфеле у него завалялся листочек с каким-то набором целых чисел, и он заинтересовался, какое из чисел меньше (в смысле обычного порядка): k -е число в наборе, отсортированном по обычным правилам, или k -е в наборе, отсортированном в смысле порядка, описанного выше (когда любое чётное число меньше любого нечётного). Помогите ему, написав программу, которая будет отвечать на этот вопрос.

Формат входа: В первой строке через пробел задано два целых числа: n — количество чисел на Петин листочке — и k — позиция чисел в отсортированных наборах ($1 \leq n \leq 10^5$, $1 \leq k \leq n$). Во второй строке через пробел задано n целых

чисел, по модулю не превосходящих 10^6 . меньшее из чисел, стоящих на k -м месте в наборах, отсортированных по указанным правилам (считаем, что первое число в наборе имеет индекс 1).

Пример 1

Вход: Выход:
3 3 5
5 100 6

Пример 2

Вход: Выход:
3 3 100
100 6 2

Примечание: В первом примере на третьем месте будет в необычном порядке будет стоять 5, поскольку в смысле необычного порядка любое нечётное число больше любого чётного; при обычной сортировке — число 100; меньшее из них — 5. Во втором примере нет нечётных чисел, поэтому обычный и необычный порядки просто совпадают и на третьем месте в обоих случаях стоит 100.

9.5. «Кратные делители». Петя Торопыжкин изучил программирование и стал решать разные математические задачи на компьютере. В частности, он вспомнил задачку из 6-го класса на понятие делимости: пусть имеется набор натуральных чисел $\{n_i\}$ и натуральное число m . Вопрос: делителем какой степени является число m для совокупного НОК чисел n_i ?

Совокупное НОК (наименьшее общее кратное) набора натуральных чисел — это наименьшее натуральное число, для которого любое число из имеющегося набора будет делителем.

Помогите Пете, напишите соответствующую программу.

Формат входа: В первой строке через пробел задано два целых числа: n — количество чисел в наборе — и число m ($1 \leq n \leq 10^5$, $2 \leq m \leq 10^6$). Во второй строке через пробел задано n натуральных чисел, не превосходящих 10^6 .

Формат выхода: Выведите единственное неотрицательное целое число — степень делителя m у совокупного НОК чисел n_i .

Пример 1

Вход: Выход:
3 2 2
100 5 2

Пример 2

Вход: Выход:
3 3 0
100 5 2

Примечание: В примерах $\text{НОК}(100, 5, 2) = 100$. В первом примере 100 делится на $4 = 2^2$, но не делится на $8 = 2^3$. Во втором примере 100 не делится на 3, то есть делится на $1 = 3^0$.

**Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2015 – 2016 учебном году
9 класс. Разбор решений и идеи тестов**

9.1. «Наибольший хвост». *Заданы три натуральных числа n_1, n_2, n_3 , не меньших 10. Выведите то из них, две последние цифры которого, будучи рассмотрены в том же порядке, что и в десятичной записи числа, дают наибольшее двузначное число.*

Задача, по мнению программного комитета, является утешительной. Для решения требуется сравнивать остатки от деления данных чисел на 100.

Идеи тестов:

- 1–3. Случайные тесты, максимум единственен.
- 4–8. Случайные тесты, два числа различны, но являются максимумом в указанном смысле.
- 9–10. Случайные тесты, три числа различны, но все являются максимумом в указанном смысле.

9.2. «Победа в бою». *В свободное от уроков и тренировок по плаванию время Петя Торопыжский играет в компьютерные игры. В очередном бою у Пети имеется отряд из k_1 пехотинцев, каждый из которых имеет силу удара s_1 и здоровье h_1 . У компьютерного противника также имеется отряд из k_2 пехотинцев, каждый с ударом s_2 и здоровьем h_2 . Отряд Пети первым наносит удар по отряду противника, затем атакует отряд противника, потом снова Петин отряд и т.д. Отряд из k человек (неважно, есть ли среди них раненый боец или нет) с ударом s наносит $k \cdot s$ единиц урона, которые последовательно вычитаются из здоровья наиболее раненного члена атакуемого отряда. Если единиц урона больше, чем здоровье такого члена отряда, то он умирает и здоровье начинает списываться у другого солдата (который также может погибнуть, если урона слишком осталось много) — до тех пор, пока есть живые солдаты или есть очки урона.*

Напишите программу, которая определяет по указанным входным данным, какой отряд выживет в таком бою и сколько солдат останется в его составе.

Вероятно, можно вывести формулу, оценивающую по входным данным игрока, чей отряд победит, и остаточную численность его отряда. Однако прямое моделирование боя является представляется более простым, поэтому данную задачу следует отнести к классу симуляторов, то есть задач, требующих аккуратной технической реализации приведённых правил изменения состояния системы. Программный комитет считает данную задачу несложной.

Как представляется программному комитету, для моделирования каждого отряда удобнее представить его в виде двух величин: здоровье w раненого бойца (величина, меньшая максимального здоровья, возможно, равная нулю, если в отряде нет раненых бойцов) и количество g полностью здоровых бойцов. Вначале $w = 0$ и $g = k$.

Пусть при очередной атаке на отряд ему наносится P очков урона. Если $P \leq w$, вычитаем из w величину P , на этом атака заканчивается. В противном случае вычитаем из P величину w и кладём $w = 0$. Результат вычитания будем обозначать P' . Если P' не меньше суммарного здоровья оставшегося отряда, то есть, если $P' \geq g \cdot h$, то отряд уничтожен полностью, бой заканчивается. Иначе вычисляем количество бойцов, убитых этой атакой $P' \div h$ и вычитаем эту величину из g . Затем, если $P' \bmod h \neq 0$, то один боец получает частичные ранения в размере $P' \bmod h$, то есть уменьшаем результирующее значение g ещё на 1 и присваиваем $w = h - P' \bmod h$. Если же $P' \bmod h = 0$, то оставляем результирующее значение g , как есть, и полагаем $w = 0$. Атака закончена, переходит к расчёту следующей атаки до тех пор, пока один из отрядов не будет полностью уничтожен.

Ответом будет величина g для оставшегося отряда, увеличенная на 1, если для этого отряда $w \neq 0$ (учитываем раненого бойца).

Идеи тестов:

1. Параметры бойцов обоих отрядов одинаковы, удар кратен здоровью, выигрывает Петин отряд.
2. Параметры бойцов отрядов неодинаковы, удары кратны здоровью бойцов противника, выигрывает Петин отряд.
3. Параметры бойцов неодинаковы, удар Петиных бойцов кратен здоровью бойцов противника, удар бойцов противника не кратен здоровью Петиных бойцов, выигрывает Петин отряд, имея только одного раненого бойца.
4. Параметры бойцов неодинаковы, удар Петиных бойцов кратен здоровью бойцов противника, удар бойцов противника не кратен здоровью Петиных бойцов, выигрывает Петин отряд, имея только одного здорового бойца.
5. Параметры бойцов неодинаковы, удар Петиных бойцов кратен здоровью бойцов противника, удар бойцов противника не кратен здоровью Петиных бойцов, выигрывает Петин отряд, имея несколько здоровых бойцов.
6. Параметры бойцов неодинаковы, удар Петиных бойцов кратен здоровью бойцов противника, удар бойцов противника не кратен здоровью Петиных бойцов, выигрывает Петин отряд, имея здоровых и раненого бойцов.
- 7–10. То же, что в тестах 3–6, только удар бойцов противника не кратен здоровью Петиных бойцов.
- 11–20. То же, что в тестах 1–10, только ситуация симметрична — выигрывает противник Пети.
- 21–25. Случайные тесты.

9.3. «Новый химический элемент». Сидя на уроке химии, Петя Торопыжский мечтал о том, как он откроет новый химический элемент. Потом он задумался о его названии и обозначении. И тут он заметил на парте в кабинете химии длинное слово, состоящее из заглавных латинских букв, и решил, что обозначением будет сочетание двух последовательных букв из увиденного слова. Но сочетаний много, и Петя решил выбрать из них минимальное в лексикографическом порядке.

Напомним, что лексикографический порядок сравнения слов — это порядок, принятый в словарях: слова сравниваются по первым буквам, при их равенстве по вторым, при равенстве вторых — по третьим и т.д., пока будет найдена несовпадающая пара букв, которая определит порядок слов, или одно из слов не кончится; в последнем случае более короткое слово, совпадающее с началом более длинного, считается меньшим.

Помогите Пете, написав программу, которая по введённому слову найдёт минимальное в лексикографическом порядке двухбуквенное подслово.

Данная задача, в каком-то смысле, походит на задачу 9.1 — здесь тоже надо найти экстремальный элемент из набора. Однако разница состоит в том, что в первой задаче набор состоит из трёх элементов и при её решении можно обойтись несколькими сравнениями, а в данной задаче необходимо реализовать алгоритм поиска экстремального элемента в наборе, размер которого заранее неизвестен. Также необходимо реализовать корректное сравнение двух двухбуквенных слов.

Пусть l — длина заданной строки s . Тогда будем задавать двухбуквенное подслово индексом i его первого символа, $1 \leq i < l$ (считаем, что индексы отсчитываются от 1). Пусть $i_{\min}$ — индекс текущего найденного минимального подслова. Вначале $i_{\min} = 1$. Перебирая индекс i текущего подслова от 2 до $l - 1$ и сравнивая подслово, задаваемое этим индексом, с текущим минимальным подсловом, задаваемым индексом $i_{\min}$, найдём минимальное подслово во всей строке.

Слово, задаваемое индексом k , меньше слова, задаваемого индексом m , если $s[k] < s[m]$ или если $s[k] = s[m]$ и $s[k + 1] < s[m + 1]$.

Идеи тестов:

1. Минимальный тест — двухбуквенная строка.
2. Два вида символов, строка чётной длины вида $ABAB \dots AB$.
3. Два вида символов, строка чётной длины вида $BABA \dots BA$.
4. Два вида символов, строка нечётной длины вида $ABAB \dots ABA$.
5. Два вида символов, строка нечётной длины вида $BABA \dots BAB$.
6. Два вида символов, случайные сочетания двух символов, нет двух меньших символов, идущих подряд.
7. Два вида символов, случайные сочетания двух символов, есть два меньших символа, идущие подряд.
- 8–11. Случайные тесты, много видов символов, нет двух меньших символов, идущих подряд, длина строки меньше 255.

- 12–14. Случайные тесты, много видов символов, есть два меньшие символа, идущие подряд, длина строки меньше 255.
- 15–17. Случайные тесты, много видов символов, нет двух меньших символов, идущих подряд, длина строки равна 255.
- 18–20. Случайные тесты, много видов символов, есть два меньшие символа, идущие подряд, длина строки равна 255.

9.4. «Странное сравнение». *На математическом кружке Петя Торпыжкин прослушал лекцию на тему отношений порядка и узнал, что целые числа можно сравнивать не только так, как это принято на уроках математики. Например, можно считать, что любое чётное число меньше любого нечётного, а пару чётных чисел или пару нечётных чисел уже сравнивать по обычным правилам — и это тоже будет порядком на натуральных числах! Правда, непривычным. В портфеле у него завалялся листочек с каким-то набором целых чисел, и он заинтересовался, какое из чисел меньше (в смысле обычного порядка): k -е число в наборе, отсортированном по обычным правилам, или k -е в наборе, отсортированном в смысле порядка, описанного выше (когда любое чётное число меньше любого нечётного). Помогите ему, написав программу, которая будет отвечать на этот вопрос.*

Собственно, прямолинейное решение данной задачи подразумевает двукратную сортировку заданного массива, каждый раз с указанным порядком: один раз с обычным, другой раз с необычным. После каждой сортировки из отсортированного массива извлекается k -е число, результаты сравниваются, выдаётся их минимум.

Таким образом, имеются две сложности. Первая состоит в том, чтобы организовать процедуру сортировки, которая работает достаточно быстро. Вторая — в том, чтобы организовать сортировку в необычном порядке. Первая сложность, в принципе, вполне по силам подготовленному школьнику: нужно просто использовать библиотечную функцию сортировки. Однако, чтобы организовать сортировку в нестандартном порядке с использованием стандартной функции сортировки требуются хорошие знания библиотеки используемого языка. Если же при этом отказаться от библиотечной функции, то необходимо умение самому писать хорошие сортировки (быструю сортировку, сортировку слиянием, сортировку кучей).

В принципе, возможно решение, опирающееся в трёх четвертях случаев на однократную сортировку массива (в необычном порядке; что, впрочем, не снимает указанные сложности, а лишь делает решение чуть более эффективным). Действительно, пусть в необычном порядке на k -м месте стоит некоторое нечётное число a . При перестановке в обычном порядке **перед** a новых чисел возникнуть не может: все чётные уже и так стояли перед a , а все нечётные, стоящие справа, больше a , и перед a встать не могут. Стало быть, a при пересортировке a может либо остаться на месте (это значит, что все чётные числа так и остались левее a ,

то есть все чётные числа меньше a), либо переместиться левее (это означает, что есть чётные числа больше a).

Пусть теперь в необычном порядке на k -м месте стоит некоторое чётное число a . При пересортировке оно не может переместиться левее: влево числа могут лишь добавляться. Стало быть, в этом случае число a может либо остаться на месте (нечётные числа левее a не становились, то есть все нечётные больше a), либо переместиться правее (то есть имеются нечётные числа, меньшие a).

Таким образом, логически возможны 4 случая:

1. в необычном порядке на k -м месте стоит некоторое нечётное число a ; все чётные числа меньше a ; искомый результат — число a ;
2. в необычном порядке на k -м месте стоит некоторое нечётное число a ; имеются чётные числа, больше a ; искомый результат — число a ;
3. в необычном порядке на k -м месте стоит некоторое чётное число a ; все нечётные числа больше a ; искомый результат — число a ;
4. в необычном порядке на k -м месте стоит некоторое чётное число a ; имеются нечётные числа, меньшие a ; искомый результат — число, стоящее на k -месте в массиве, отсортированном в обычном порядке.

Таким образом, видно, что если в необычном порядке на k -м месте стоит нечётное число a , то оно же будет результатом. В случае, когда это число чётное, оно будет ответом, если нет нечётных, меньших его. Этот факт можно установить за один проход массива. И, наконец, если число a чётное и имеются нечётные, меньшие a , то требуется сортировка в смысле обычного порядка чисел.

Идеи тестов:

1. Минимальный тест: $n = k = 1$. Единственное число — чётное.
2. Минимальный тест: $n = k = 1$. Единственное число — нечётное.
3. Минимальный тест: $n = 2, k = 2$, ответ — большее из этих двух чисел (в обычном порядке).
4. Минимальный тест: $n = 2, k = 2$, ответ — меньшее из этих двух чисел (в обычном порядке).
5. Минимальный тест: $n = 2, k = 1$, ответ, соответственно — меньшее из этих двух чисел (в обычном порядке).
6. $n = 20$, нет нечётных чисел.
- 7–10. $n = 20$, есть нечётные числа, рассматриваются случаи 1–4.
- 11–15. То же, что в тестах 6–10, только $n = 5000$.
- 16–20. То же, что в тестах 6–10, только $n = 10000$.
- 21–25. То же, что в тестах 6–10, только $n = 100000$.

9.5. «Кратные делители». *Петя Торопыжский изучил программирование и стал решать разные математические задачи на компьютере. В частности,*

он вспомнил задачку из 6-го класса на понятие делимости: пусть имеется набор натуральных чисел $\{n_i\}$ и натуральное число m . Вопрос: делителем какой степени является число m для совокупного НОК чисел n_i ?

Совокупное НОК (наименьшее общее кратное) набора натуральных чисел — это наименьшее натуральное число, для которого любое число из имеющегося набора будет делителем.

Помогите Пете, напишите соответствующую программу.

Задача идейно несложна, однако полное её решение не вполне прямолинейно, а также для своей реализации требует определённого уровня программирования, что может вызвать трудности у девятиклассников.

Далее в тексте $\sigma(b, a)$ — степень делителя b у числа a . Алгоритм вычисления этой величины достаточно прост:

```
 $\sigma := 0;$   
while  $a \bmod b = 0$  do begin  
  inc( $\sigma$ );  
   $a := a \bmod b$ ;  
end;
```

Единственно, при этом разрушается значение a , поэтому этот код разумно оформить как функцию.

Прямолинейное решение состоит в нахождении НОК всех предложенных чисел a_i и в последующем подсчете степени указанного делителя у найденного НОК. Недостаток предложенного метода заключается в том, что НОК может оказаться настолько большим, что не войдёт в тип `int`. Те, кто пишут на Java или на Python, могут привлечь библиотеки длинной арифметики. Но на самом деле, этого не требуется.

Решение основано на элементарном наблюдении, что степень того или иного простого делителя у НОК не может быть больше степени этого делителя у чисел, НОК которых подсчитывается. Однако это, вообще говоря, неверно для составных делителей. Действительно, 63 не является делителем ни 7, ни 9, но является делителем их НОК, равного как раз 63. То есть сомножители, обеспечивающие тот факт, что m является делителем НОК, могут «приходить» из разных чисел, у которых это НОК вычисляется.

Стало быть, надо вычислить степени простых делителей m , встречающихся чисел a_i . Идея оптимального алгоритма состоит в следующем:

1) раскладываем m на простые множители, запоминая значение каждого множителя p_j и его степень $d_j = \sigma(p_j, m)$; отметим, что при ограничении $m \leq 10^6$ число m не может иметь более 8 различных простых делителей: 2, 3, 5, 7, 11, 13, 17, 19 — их произведение уже больше 10^6 ; если брать следующие простые числа, то миллион превысит произведение меньшего их количества;

2) последовательно считываем предложенные числа a_i , для каждого простого делителя p_j числа m подсчитываем его степень $g_{i,j} = \sigma(p_j, a_i)$ у каждого из

чисел a_i и находим максимум $g_j^* = \max_i g_{i,j}$ из величин $g_{i,j}$ для каждого j ;

3) понятно, что если m является делителем степени k у какого-то числа l , то каждый простой делитель p_j числа m должен быть делителем числа l степени не ниже $k \cdot d_j$, то есть $k \leq \sigma(p_j, l) \operatorname{div} d_j$, и при этом хотя бы один делитель $p_{\bar{j}}$ является делителем степени ниже $(k+1) \cdot d_{\bar{j}}$. Отсюда вытекают действия заключительного этапа: ищем минимум $k^* = \min_j (g_j^* \operatorname{div} d_j)$ — это и будет ответ.

Идеи тестов:

1. Минимальный тест: $n = 1$, заданное число взаимно просто с m .
2. Минимальный тест: $n = 1$, заданное число имеет m делителем степени 1.
3. Минимальный тест: $n = 1$, заданное число имеет m делителем степени больше 1.
4. $n = 25$, $m = p_1 \cdot p_2$ есть произведение двух простых сомножителей, m взаимно просто с НОК n_i .
5. $n = 25$, $m = p_1 \cdot p_2$, степень делителя m у НОК есть максимум степеней делителя m у чисел n_i .
6. $n = 25$, $m = p_1 \cdot p_2$, m не является делителем ни одного n_i , одно из чисел из набора содержит в качестве делителя один из простых сомножителей m , другое — другой.
- 7–9. То же, что в тестах 4–6, только НОК не входит в `int`.
10. $n = 25$, $m = p_1^{k_1} \cdot p_2^{k_2} \cdot p_3^{k_3}$, m взаимно просто с НОК n_i .
11. $n = 25$, $m = p_1^{k_1} \cdot p_2^{k_2} \cdot p_3^{k_3}$, степень делителя m у НОК есть максимум степеней делителя m у чисел n_i .
12. $n = 25$, $m = p_1^{k_1} \cdot p_2^{k_2} \cdot p_3^{k_3}$, m не является делителем ни одного n_i , n_i «приносят» в НОК два делителя m .
13. $n = 25$, $m = p_1^{k_1} \cdot p_2^{k_2} \cdot p_3^{k_3}$, m не является делителем ни одного n_i , n_i «приносят» в НОК по одному делителю m .
- 14–17. То же, что в тестах 10–13, только $n = 50$ и НОК не входит в `int`.
- 18–21. То же, что в тестах 10–13, только $n = 50$ и НОК не входит в `int64`.
- 22–33. То же, что в тестах 10–21, только $n = 50$, $m = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17$.
- 34–41. Случайные тесты, НОК входит в тип `int`. Есть один максимальный тест $n = 10^5$.
- 42–46. Случайные тесты, НОК не входит в тип `int`. Есть один максимальный тест $n = 10^5$.
- 47–50. Случайные тесты, НОК не входит в тип `int64`. Есть один максимальный тест $n = 10^5$.

Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2015 – 2016 учебном году
10 класс

Время выполнения задач — 4 часа

Ограничение по времени — 2 секунды на тест

Ограничение по памяти — 64 мегабайта

10.1. «Сумма последовательности». Формула общего члена последовательности —

$$a_n = \text{round}(|10 \cdot \sin(n^2 + 2n + 5)|).$$

Здесь аргумент синуса подразумевается в радианах, $|\cdot|$ — функция модуля числа, $\text{round}(\cdot)$ — функция округления. Для заданного натурального числа S найдите минимальный номер k такой, что сумма последовательности $a_1 + a_2 + \dots + a_k$ превзойдёт S .

Формат входа: На входе задаётся единственное натуральное число — величина S ($1 \leq S \leq 1000$).

Формат выхода: Выдайте единственное натуральное число k — искомый индекс.

Пример

<u>Вход:</u>	<u>Выход:</u>
22	3

Примечание: Указанная последовательность имеет вид 10, 4, 9, 7, ..., поэтому сумма первых трёх её членов превзойдёт 22.

10.2. «Вариация массива». Петя Торопыжкин назвал *вариацией массива* следующую операцию: массив сортируется по возрастанию элементов, затем находится разность второго и первого элементов отсортированного массива, затем третьего и второго, затем четвёртого и третьего, и т.д. до разности последнего и предпоследнего элементов. После все найденные разности суммируются. Помогите Пете, написав программу, которая проделывает указанную операцию над заданным массивом.

Формат входа: В первой строке задано целое число n — количество элементов в массиве ($2 \leq n \leq 1000$). В следующей строке через пробел задано n целых чисел, каждое по модулю не превосходит 10^6 .

Формат выхода: Выведите единственное целое число — вариацию заданного набора чисел.

Пример

<u>Вход:</u>	<u>Выход:</u>
4	5
5 2 6 1	

Примечание: Результат получается следующим образом: отсортированный массив: 1 2 5 6; вычисляем разности: $2 - 1 = 1$, $5 - 2 = 3$, $6 - 5 = 1$; вычисляем сумму разностей: $1 + 3 + 1 = 5$.

10.3. «Космическая стратегия». В одной космической стратегической игре, которая нравится Пете Торопыжкину, производство на планете зависит от величины её населения. Начав новую партию, Петя собрал информацию о численности населения каждой из планет, присутствующих в игре, и хочет выяснить, какая численность населения является наиболее распространённой в игровой Вселенной (чтобы знать, на какой уровень производства можно рассчитывать в среднем). Помогите ему, напишите соответствующую программу.

Формат входа: В первой строке задано целое число n — количество планет в игровой Вселенной ($1 \leq n \leq 10^5$). В следующей строке через пробел в каком-то порядке задано n натуральных чисел, не превосходящих 10^9 — населения планет.

Формат выхода: Выведите единственное натуральное число — численность населения, наиболее часто встречающуюся в текущей Вселенной. Если таких численностей несколько, выведите меньшую из них (чтобы Пете готовиться к худшему из возможных вариантов).

Пример 1

Вход:

4

100 50 100 5

Выход:

100

Пример 2

Вход:

4

100 50 100 50

Выход:

50

10.4. «Корень сравнения». На математическом кружке Петя Торопыжкин изучил тему «Сравнения». В теории чисел *алгебраическим сравнением по модулю p* (p — натуральное число) называют выражение вида

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_2 x^2 + a_1 x + a_0 \equiv 0 \pmod{p},$$

то есть выражение, в левой части которого стоит многочлен с целыми коэффициентами. Корнем сравнения называют такое целое число, которое, будучи подставленным в левую часть, даёт значение, *сравнимое с нулём по модулю p* , то есть дающее остаток 0 при делении на p . Понятно, что если сравнение имеет один корень x_0 , то корнями являются также числа $x_0 + k \cdot p$, k — любое целое число.

Пете задали несколько сравнений на дом, и он решил все задачи, кроме одной, в которой он несколько коэффициентов многочлена в левой части записал неразборчиво из-за кончающейся ручки. Тогда он задался вопросом: каковы могут быть значения этих коэффициентов, чтобы число $p - 1$ было корнем этого сравнения? Опять понятно, что если какое-то число a_i является ответом к задаче, которую поставил себе Петя, то числа $a_i + k \cdot p$ тоже будут ответами. Поэтому Петя хочет найти ответ, в котором все неопределённые коэффициенты лежат в диапазоне от 0 до $p - 1$ включительно. Помогите ему, напишите соответствующую программу. Заметим, что такая задача всегда имеет решение, если хотя бы

один из коэффициентов неопределён. Также считаем, что старший коэффициент заведомо является определённым.

Формат входа: В первой строке перечислены три натуральных числа: n — степень сравнения (степень многочлена в левой части), k — количество неопределённых коэффициентов — и p — модуль сравнения ($1 \leq n \leq 1000$, $1 \leq k \leq n$, $2 \leq p \leq 10^6$). В следующей строке через пробел в порядке убывания перечислены степени переменной x , при которых коэффициенты не определены — целые числа из диапазона от 0 до $n - 1$. В третьей строке через пробел перечислены $(n + 1 - k)$ чисел — заданные коэффициенты в порядке убывания степеней, целые числа, по модулю не превышающие 10^6 .

Формат выхода: Выдайте набор из k целых чисел, каждое из диапазона от 0 до $p - 1$, таких, что при их подстановке в сравнение вместо неопределённых коэффициентов (в порядке убывания степеней) полученное сравнение будет иметь корень $p - 1$. Если ответов несколько, выдайте любой из них.

Пример

Вход: Выход:

3 2 5 0 4
2 0
4 -10

Примечание: В примере задано сравнение $4x^3 + ?x^2 - 10x + ? \equiv 0 \pmod{5}$ ($?$ — неопределённые коэффициенты). Подстановка вместо неопределённых коэффициентов чисел из предлагаемого ответа (0 в коэффициент у x^2 , 4 вместо свободного члена) даёт сравнение $4x^3 - 10x + 4 \equiv 0 \pmod{5}$, которое действительно имеет корень $4 = 5 - 1$: $4 \cdot 4^3 - 10 \cdot 4 + 4 = 256 - 40 + 4 = 220 \equiv 0 \pmod{5}$.

10.5. «Странное сравнение строк». Петя Торопыжкин решил придумать новый способ сравнения строк: фиксируется строка-ключ, и рассматривается количество вхождений сравниваемых строк в строку-ключ. Если эти количества не равны, то они определяют порядок строк, если же они равны, то строки сравниваются в лексикографическом порядке. Петя решил написать программу, которая для заданной строки-ключа находит наибольшую строку из заданного набора в смысле придуманного им порядка. Помогите ему в написании такой программы.

Формат входа: В первой строке задана строка-ключ. Во второй строке задано целое число n — количество строк в наборе ($1 \leq n \leq 10000$). В следующих n строках заданы строки из набора. Каждая строка (из набора и строка-ключ) задаётся в виде `длина_символы`. Все строки непусты, состоят из заглавных букв латиницы. Строка-ключ имеет длину не более 10000 символов, строки из набора — не более 100 символов. Все строки из набора попарно различны.

Формат выхода: Выдайте единственную строку — наибольшую в смысле предложенного Петей порядка.

Пример 1

Вход: Выход:

8 ABCABCAВ АВ

3

2 АВ

3 BCD

2 BC

Пример 2

Вход: Выход:

9 ABCABCAВC BC

3

2 АВ

3 BCD

2 BC

**Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2015 – 2016 учебном году
10 класс. Разбор решений и идеи тестов**

10.1. «Сумма последовательности». *Формула общего члена последовательности —*

$$a_n = \text{round}(|10 \cdot \sin(n^2 + 2n + 5)|).$$

Здесь аргумент синуса подразумевается в радианах, $|\cdot|$ — функция модуля числа, $\text{round}(\cdot)$ — функция округления. Для заданного натурального числа S найдите минимальный номер k такой, что сумма последовательности $a_1 + a_2 + \dots + a_k$ превзойдёт S .

Задача имеет статус утешительной и подразумевает прямолинейное суммирование слагаемых, вычисляемых по заданной формуле. Некоторая сложность состоит в том, что язык, используемый участником, не поддерживает операцию округления. Её замена: $\text{round}(x) = [x + 0.5]$, где $[\cdot]$ — операция отбрасывания целой части.

Идеи тестов:

1. Минимальный тест: $S = 1$.
2. Минимальный тест: $S = 9$.
- 3–10. Случайные тесты для S в диапазоне от 50 до 1000.

10.2. «Вариация массива». *Петя Торопыжкин назвал вариацией массива следующую операцию: массив сортируется по возрастанию элементов, затем находится разность второго и первого элементов отсортированного массива, затем третьего и второго, затем четвёртого и третьего, и т.д. до разности последнего и предпоследнего элементов. После все найденные разности суммируются. Помогите Пете, написав программу, которая проделывает указанную операцию над заданным массивом.*

Данная задача представляет два различных пути решения, один из которых весьма прост в реализации. Прямолинейное решение подразумевает считывание элементов массива, его сортировку, а затем нахождение и суммирование разностей.

Однако, если задуматься о сути проделываемой операции, то можно понять, что искомый результат можно найти без применения операции сортировки. Действительно, пусть $a_1, a_2, \dots, a_n$ — последовательность элементов в массиве, отсортированном по возрастанию. При этом a_1 — минимальный элемент, а a_n — максимальный. Тогда искомая величина есть

$$\begin{aligned} S &= (a_2 - a_1) + (a_3 - a_2) + (a_4 - a_3) + \dots + (a_{n-1} - a_{n-2}) + (a_n - a_{n-1}) = \\ &= a_2 - a_1 + a_3 - a_2 + a_4 - a_3 + \dots + a_{n-1} - a_{n-2} + a_n - a_{n-1} = a_n - a_1. \end{aligned}$$

Последнее равенство получаем потому, что в выражении в последней строке все слагаемые, кроме a_1 и a_n , встречаются дважды с разными знаками и взаимно уничтожаются.

Стало быть, для получения искомой величины надо найти разность максимального и минимального элементов из заданного набора, которые ищутся стандартными алгоритмами без сортировки массива.

Идеи тестов:

1. Все элементы набора одинаковы.
- 2–6. Случайные тесты, где величины a_i по модулю не превосходят 1000 и все промежуточные результаты при прямолинейном вычислении гарантированно входят в тип `short int`.
- 7–10. Случайные тесты, где величины a_i большие и промежуточные вычисления могут не войти в тип `short int`.

10.3. «Космическая стратегия». *В одной космической стратегической игре, которая нравится Пете Торопыжскину, производство на планете зависит от величины её населения. Начав новую партию, Петя собрал информацию о численности населения каждой из планет, присутствующих в игре, и хочет выяснить, какая численность населения является наиболее распространённой в игровой Вселенной (чтобы знать, на какой уровень производства можно рассчитывать в среднем). Помогите ему, напишите соответствующую программу.*

Данная задача предполагается имеющей среднюю сложность. Фактически, требуется найти элемент, входящий в данный набор максимальное количество раз.

Прямолинейное решение, основанное на подсчёте количества вхождений каждого числа в заданный набор, может потребовать либо слишком большой памяти (массив на 10^9 элементов), либо слишком большого времени — если последовательно подсчитывать количество вхождений 1, затем 2, затем 3 и т.д. Впрочем, этот путь можно реализовать при помощи специальной структуры данных — словаря (ассоциативного массива). Он позволит хранить записи количеств не всех чисел от 1 до 10^9 , а лишь тех, которые реально присутствуют в наборе. Количество их не превосходит размера набора, то есть 10^5 , что и уложится в допустимую память, и будет достаточно быстрым по времени. Однако такая структура данных не присутствует в библиотеке языка Паскаль в средах Delphi и FreePascal/Lazarus, а её самостоятельное написание составляет определённую сложность. Однако словарь присутствует в библиотеке языков Pascal.ABC, C++, Java и Python.

Альтернативный путь решения подразумевает предварительную сортировку массива по возрастанию, что приведёт к группировке одинаковых элементов — они будут идти в массиве подряд. Пусть a_i — i -й элемент в отсортированном массиве, $1 \leq i \leq n$. Затем нужно найти элемент, группа которого имеет наибольшую длину. Для реализации этого алгоритма будем хранить величины `elMax` —

величину элемента, максимально входящего при обработке массива до текущей позиции, и `qntMax` — количество его вхождений. Вначале положим `elMax = 0`, `qntMax = -1`. Кроме того, понадобится длина текущей обрабатываемой группы `qnt`; вначале положим эту величину равной 1 (первый элемент массива даёт группу длины, по крайней мере, 1). Затем организуем цикл по счётчику i от 2 до n . (Если в наборе всего один элемент, то есть если $n = 1$, ответ тривиален.) Если $a_i = a_{i-1}$, то мы находимся в рамках группы одного элемента, увеличиваем `qnt` на 1. В противном случае `qnt` есть длина группы элемента a_{i-1} ; сравниваем `qnt` и `qntMax`; если `qnt > qntMax`, то присваиваем `qntMax = qnt`, `elMax = ai-1`; затем присваиваем `qnt = 1` — длина текущей рассмотренной части группы элемента a_i . Заметим, что строгое сравнение `qnt` и `qntMax` гарантирует сохранение меньшего элемента в качестве ответа при равенстве количеств вхождений, поскольку меньший элемент будет обработан раньше. После окончания цикла надо аналогично сравнить накопленное значение `qnt`, которое описывает количество вхождений элемента a_n .

Идеи тестов:

1. Минимальный тест: $n = 1$.
- 2–11. Случайные тесты, элемент с максимальным количеством вхождений единственен. Имеется максимальный тест $n = 10^5$.
- 12–20. Случайные тесты, несколько различных элементов с одинаковым максимальным количеством вхождений. Имеется максимальный тест $n = 10^5$.

10.4. «Корень сравнения». На математическом кружке Петя Торопыжский изучил тему «Сравнения». В теории чисел алгебраическим сравнением по модулю p (p — натуральное число) называют выражение вида

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_2 x^2 + a_1 x + a_0 \equiv 0 \pmod{p},$$

то есть выражение, в левой части которого стоит многочлен с целыми коэффициентами. Корнем сравнения называют такое целое число, которое, будучи подставленным в левую часть, даёт значение, сравнимое с нулём по модулю p , то есть дающее остаток 0 при делении на p . Понятно, что если сравнение имеет один корень x_0 , то корнями являются также числа $x_0 + k \cdot p$, k — любое целое число.

Пете задали несколько сравнений на дом, и он решил все задачи, кроме одной, в которой он несколько коэффициентов многочлена в левой части записал неразборчиво из-за кончающейся ручки. Тогда он задался вопросом: каковы могут быть значения этих коэффициентов, чтобы число $p - 1$ было корнем этого сравнения? Опять понятно, что если какое-то число a_i является ответом к задаче, которую поставил себе Петя, то числа $a_i + k \cdot p$ тоже будут ответами. Поэтому Петя хочет найти ответ, в котором все неопределённые коэффициенты лежат в диапазоне от 0 до $p - 1$ включительно. Помогите ему, напишите

соответствующую программу. Заметим, что такая задача всегда имеет решение, если хотя бы один из коэффициентов неопределён. Также считаем, что старший коэффициент заведомо является определённым.

Решение данной задачи технически не очень сложно, но для формулировки корректного алгоритма требуются определённые размышления из области теории чисел.

В описании решения подразумевается, что все вычисления проделываются по модулю p . При этом следует учесть, что чаще всего в языках программирования функция взятия остатка реализована таким образом, что выдаёт результат, имеющий знак делимого, в то время как традиционное математическое определение этой операции подразумевает неотрицательный результат в диапазоне от 0 до величины, на 1 меньшей делителя. Собственно, коррекция результата весьма проста: если результат получился отрицательным, к нему надо прибавить делитель.

Решение базируется на следующих наблюдениях.

1. Если величина d взаимно проста с p (то есть $\text{НОД}(d, p) = 1$), то значение выражения $(d \cdot x) \bmod p$ при переборе значений x от 0 до $p - 1$ даёт все значения из диапазона от 0 до $p - 1$. Это обеспечит существование решения.

2. Величина $p - 1$, будучи возведённой в любую неотрицательную степень, взаимно проста с p .

Отсюда путь решения заключается в следующем. Полагаем равными нулю все коэффициенты кроме одного. Пусть это будет коэффициент при некотором слагаемом с x^k , $k \geq 0$. Вычисляем сумму s всех слагаемых с определёнными уже коэффициентами. Вычисляем $d = (p - 1)^k \bmod p$. В результате имеем сравнение $s + a_k \cdot d \equiv 0 \pmod{p}$, которое нужно решить относительно коэффициента a_k .

Некоторое рассуждение относительно вычисления s и d . Потенциально нам нужно умножать два числа, модули которых заключены в диапазоне от 0 до $p - 1$, то есть в худшем случае до 10^6 : при вычислении $(p - 1)^j$ надо перемножать величины $p - 1$, а потом ещё умножать на коэффициент. Если делать это прямолинейно, то возможно переполнение типа `int`: произведение может иметь модуль до 10^{12} , которое только потом с использованием операции остатка переводится в диапазон от 0 до $p - 1$. Для избежания этого можно использовать более значащий тип `int64` или (для тех, кто пишет на Java или Python) применять длинную арифметику. Однако следующее математическое соображение упростит этот момент. Величина $(p - 1)$ сравнима с (-1) по модулю p , значит, $(p - 1)^2$ сравнима с 1, $(p - 1)^3$ снова сравнима с (-1) и т.д. Вообще, $(p - 1)^{2k+1} \equiv (-1) \pmod{p}$, $(p - 1)^{2k} \equiv 1 \pmod{p}$. То есть вычисление остатков от деления $(p - 1)^i$ на p вообще не требует умножений, а умножение $(p - 1)^i$ на коэффициент не увеличивает разрядность результат и заключается лишь, возможно, в смене знака коэффициента.

После вычисления s и d имеем сравнение $s + a_k \cdot (-1)^{k \bmod 2} \equiv 0 \pmod{p}$. Откуда $a_k = (-1)^{k \bmod 2 + 1} \cdot s$. Последнюю величину нужно привести к диапазону от 0 до $p - 1$.

Идеи тестов:

1. Линейное сравнение, не определён свободный член, $p = 10$.
2. Линейное сравнение, не определён свободный член, $p = 10^6$.
3. Квадратное сравнение, не определён свободный член, $p = 10$.
4. Квадратное сравнение, не определён второй коэффициент, $p = 10$.
5. Квадратное сравнение, не определены второй коэффициент и свободный член, $p = 10$.
- 6–8. То же, что в тестах 3–5, только $p = 10^6$.
9. Максимальный тест: $n = 1000$, коэффициенты ± 999999 , не определён свободный член, $p = 10^6$.
10. Максимальный тест: $n = 1000$, коэффициенты $\pm 10^6$, не определён свободный член, $p = 10^6$.
11. Максимальный тест: $n = 1000$, коэффициенты ± 999999 , не определено около половины коэффициентов, $p = 10^6$.
12. Максимальный тест: $n = 1000$, коэффициенты $\pm 10^6$, не определено около половины коэффициентов, $p = 10^6$.
13. Максимальный тест: $n = 1000$, коэффициенты ± 999999 , не определены все коэффициенты кроме старшего, $p = 10^6$.
14. Максимальный тест: $n = 1000$, коэффициенты $\pm 10^6$, не определены все коэффициенты кроме старшего, $p = 10^6$.
- 15–20. Случайные тесты: $n \in [10, 20]$, модули всех определённых коэффициентов из диапазона $[-30, 30]$, $p \in [10, 100]$.
- 21–25. Случайные тесты: $n \in [900, 1000]$, модули всех определённых коэффициентов из диапазона $[-10^6, 10^6]$, $p \in [900000, 10^6]$.

10.5. «Странное сравнение строк». *Петя Торопыжский решил придумать новый способ сравнения строк: фиксируется строка-ключ, и рассматривается количество вхождений сравниваемых строк в строку-ключ. Если эти количества не равны, то они определяют порядок строк, если же они равны, то строки сравниваются в лексикографическом порядке. Петя решил написать программу, которая для заданной строки-ключа находит наибольшую строку из заданного набора в смысле придуманного им порядка. Помогите ему в написании такой программы.*

По мнению программного комитета, данная задача имеет высокую сложность, поскольку требует как достаточно хорошей техники программирования даже для реализации частичного решения, так и знания алгоритмов работы со строками.

В основе решения задачи лежит алгоритм поиска максимума среди заданного набора элементов. Идея его проста: в качестве начального значения максимума положим первый элемент набора. Если очередной считанный элемент больше текущего максимума, запоминаем его в качестве нового найденного максимума. Проблема состоит в том, что функция сравнения требует вычислить количество

вхождений сравниваемых строк в строку-ключ. Для текущего максимума эта информация запомнена, а вот для каждой новой строки эту величину требуется посчитать.

Обозначим через S строку-ключ, а через T — очередную обрабатываемую строку из набора. При реализации простейшего подхода к подсчёту количества вхождений строки T в строку S алгоритм перебирает все позиции строки S и смотрит, не входит ли T в S , начиная с этой позиции. То есть для каждой позиции в S нам нужно сделать сравнений символов в худшем случае столько, какова длина T . Стало быть, сложность обработки одной строки T есть $O(|S| \cdot |T|)$ (модуль означает длину строки). В целом же для обработки набора из n строк потребуется время $O(n \cdot |S| \cdot |T|)$, что для имеющихся ограничений слишком долго.

Для быстрого подсчёта количества (и поиска) вхождений строки T в строку S за время $O(|S| + |T|)$ может быть применён алгоритм Кнута – Морриса – Пратта, основывающийся на понятии *префикс-функции* строки. Значение префикс-функции $\pi(D, i)$ для строки D и индекса $i = 2, \dots, |D|$, есть наибольшая длина начальной части подстроки $D(1..i)$ (то есть *префикса* строки $D(1..i)$), которая является концом (*суффиксом*) этой подстроки (так называемый *префикс-суффикс*). Иногда для удобства считают $\pi(D, 1) = 0$. В качестве классического примера рассматривают префикс-функцию для строки $D = ABCDABSCABCDABIA$ (длина которой равна 16):

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi(D, i)$	0	0	0	0	1	2	0	0	1	2	3	4	5	6	0	1

Действительно, для значения индекса $i = 3$ рассматривается префикс ABC , не имеющий непустого префикс-суффикса. А для значения индекса $i = 12$ рассматривается префикс $ABCDABSCABCD$, который имеет префикс-суффикс $ABCD$ длины 4.

При значении i , равном длине $|D|$ строки D , рассматривают значение $\pi(D, |D|)$, меньшее $|D|$.

Алгоритм вычисления набора значений префикс-функции для всех возможных значений индекса i линеен по размеру строки и основан на принципе динамического программирования. Дальше символ d_i будет обозначать i -й символ строки D . База метода имеется: $\pi(D, 1) = 0$. Пусть $\pi(D, i) = k$. Если $d_{i+1} = d_{k+1}$ (то есть $(i+1)$ -й символ D совпадает с $(k+1)$ -м символом), то имеется префикс-суффикс длины $k+1$, положим $\pi(D, i+1) = k+1$. Пусть теперь $d_{i+1} \neq d_{k+1}$. Имеем следующую схему:

$$D = \underline{d_1 d_2 \dots d_{k-1} d_k} d_{k+1} \dots d_{i-k} \underline{d_{i-k+1} d_{i-k+2} \dots d_{i-1} d_i} d_{i+1} \dots$$

(подчёркнуты совпадающие подстроки длины k). Стало быть, имеем $d_1 = d_{i-k+1}$, $d_2 = d_{i-k+2}$, $\dots$, $d_{k-1} = d_{i-1}$, $d_k = d_i$, а $d_{k+1} \neq d_{i+1}$. Нужно найти такой более короткий префикс $D(1..j)$ строки D некоторой длины j , чтобы, во-первых, он был суффиксом подстроки $D(1..i)$, а во-вторых, $d_{j+1} = d_{k+1}$:

$$D = \overbrace{d_1 d_2 \dots d_{j-1} d_j} \overbrace{d_{j+1} \dots d_{k-1} d_k} \overbrace{d_{k+1} \dots d_{i-k} d_{i-k+1} d_{i-k+2} \dots d_{i-j+1} d_{i-j+2} \dots d_{i-1} d_i} d_{i+1} \dots$$

Верхними фигурными скобками выделены подстроки, которые по первому свойству должны совпадать.

Из новой схемы имеем, что

$$d_j = d_i = d_k, \quad d_{j-1} = d_{i-1} = d_{k-1}, \quad d_{j-2} = d_{i-2} = d_{k-2}, \quad \dots, \\ d_2 = d_{i-j+2} = d_{k-j+2}, \quad d_1 = d_{i-j+1} = d_{k-j+1}.$$

Здесь вторые равенства обеспечиваются совпадением подчёркнутых подстрок. Отсюда имеем, что подстрока $D(1..j)$ является префикс-суффиксом подстроки $D(1..k)$:

$$D = \overbrace{d_1 d_2 \dots d_{j-1} d_j} d_{j+1} \dots d_{k-j} \overbrace{d_{k-j+1} d_{k-j+2} \dots d_{k-1} d_k} \dots$$

А максимальная длина префикс-суффикса подстроки $D(1..k)$ есть $\pi(D, k)$. Стало быть, для проверки сначала надо взять $j = \pi(D, k)$. Если при этом $d_{j+1} = d_{i+1}$, то положить $\pi(D, i+1) = j+1 = \pi(D, k) + 1$. Если же $d_{j+1} \neq d_{i+1}$, то надо снова укоротить строку, и в силу аналогичных рассуждений надо взять $j' = \pi(D, j)$, и т.д., пока не будет найден индекс $j^* > 0$ такой, что $d_{j^*+1} = d_{i+1}$, либо не получим, что очередной индекс $j^* = 0$. В первом случае полагаем $\pi(D, i) = j^* + 1$, в последнем — $\pi(D, i+1) = 0$.

В целом, алгоритм построения набора $p[]$ значений префикс-функции $\pi(D, \cdot)$ строки D выглядит следующим образом:

```
p[1] := 0;
k := 0;
for i := 1 to |D|-1 do
begin
  while (k > 0) and (D[k+1] <> D[i+1]) do k := p[k];
  if D[k+1] = D[i+1] then inc(k);
  p[i+1] := k;
end;
```

Первая строка в цикле осуществляет (при необходимости) укорочение рассматриваемого префикса. Вторая строка увеличивает на 1 длину найденного префикс-суффикса (если он найден). Третья строка запоминает найденное значение. Сложность этого алгоритма есть $O(n)$, где n — длина обрабатываемой строки. Это, кстати, весьма тонкий вопрос — почему внутренний цикл `while` не добавляет сложности. За доказательством этого факта отправим читателя к книге Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л., Штайн К., Алгоритмы. Построение и анализ. М.: Вильямс, 2013.

Пусть мы умеем строить префикс-функцию заданной строки. Вернёмся к задаче подсчёта вхождений строки T в строку S . Алгоритм Кнута – Морриса – Пратта

базируется на следующей идее. Рассмотрим строку $D = T\#S$ (символ $\#$ такой, что он гарантированно не входит ни в T , ни в S ; в рамках данной задачи может быть взят любой небуквенный символ) и станем строить префикс-функцию такой строки. Если для какого-либо индекса i^* из диапазона $|T|+2 \dots |T|+|D|+1$ (то есть из диапазона, соответствующего символам строки S) окажется, что $\pi(D, i^*) = |T|$, то мы нашли место окончания префикс-суффикса длины $|T|$, то есть префикс-суффикса, совпадающего с T , то есть нашли последний символ вхождения подстроки T . Сложность данного алгоритма есть $O(|T| + |S|)$ — время построения префикс-функции для объединенной строки.

Таким образом, общая схема достаточно эффективного решения данной задачи выглядит следующим образом. Считываем строку-ключ S . В цикле считываем строки из набора. Для каждой очередной строки T алгоритмом Кнута – Морриса – Пратта считаем количество её вхождений в строку-ключ (здесь разумно реализовать функцию, которая будет возвращать символ из объединённой строки $T\#S$ по его индексу). Имея эту информацию, отрабатываем алгоритм поиска наибольшего элемента из заданного набора по функции сравнения

$$\text{str}_1 \preceq \text{str}_2 \Leftrightarrow E(\text{str}_1) < E(\text{str}_2) \vee (E(\text{str}_1) = E(\text{str}_2) \wedge \text{str}_1 \leq \text{str}_2).$$

Символ $\preceq$ означает «меньше или равно в смысле Петинского порядка», $E(\cdot)$ — количество вхождения данной строки в строку-ключ, $\leq$ — сравнение строк в обычном лексикографическом порядке.

Сложность такого алгоритма — $O(n(|S| + |T|))$, где $|S|$ — длина строки-ключа, $|T|$ — (максимальная) длина строки из набора, n — количество строк в наборе. Заметим, что существует ещё более эффективный алгоритм со сложностью $O(|S| + n|T|)$ (алгоритм Ахо – Корасик), однако его реализация значительно более трудоёмка нежели реализация алгоритма Кнута – Морриса – Пратта.

Отметим, что участники, использующие язык Python, при решении данной задачи имеют некоторое преимущество: в стандартной библиотеке этого языка для работы со строками уже реализована быстрая функция подсчёта количества вхождений данной строки в другую.

Далее в идеях тестов L — длина строки-ключа, l — длины строк из набора.

Идеи тестов:

1. Минимальный тест: $L = l = 1$, $n = 5$, у всех строк вхождение 0.
2. Минимальный тест: $L = l = 1$, $n = 5$, есть строка с вхождением 1.
3. $L = 2$, $l = 1$, $n = 5$, у всех строк вхождение 0.
4. $L = 2$, $l = 1$, $n = 5$, у двух строк вхождение 1.
5. $L = 2$, оба символа одинаковы, $l \leq 2$, $n = 5$, есть строка с вхождением 2 и вхождением 1.
6. $L = 20$, $l \leq 7$, $n = 5$, есть строка с вхождением 5, вхождения не пересекаются.

7. $L = 20$, $l \leq 7$, $n = 5$, есть строка с вхождением 5, вхождения пересекаются.
8. $L = 20$, $l \leq 7$, $n = 5$, есть две строки с вхождением 5, вхождения пересекаются.
- 9–12. Случайные тесты: $L = 100$, $l \in [10, 20]$, $n = 50$; строки могут иметь префикс, являющийся подстрокой ключа.
- 13–16. Случайные тесты: $L = 1000$, $l \in [40, 60]$, $n = 500$; строки могут иметь префикс, являющийся подстрокой ключа.
- 17–19. Случайные тесты: $L \in [5000, 8000]$, $l \in [70, 100]$, $n = 1000$; строки могут иметь префикс, являющийся подстрокой ключа.
20. $L = 100$, строка состоит из одинаковых символов, $l \in [1, 15]$, $n = 20$, есть строки разной длины, состоящие из одного символа того же, что и строка-ключ; прочие строки могут иметь префикс, являющийся подстрокой ключа.
21. $L = 1000$, строка состоит из одинаковых символов, $l \in [1, 50]$, $n = 200$, есть строки разной длины, состоящие из одного символа того же, что и строка-ключ; прочие строки могут иметь префикс, являющийся подстрокой ключа.
22. $L = 10000$, строка состоит из одинаковых символов, $l \in [1, 100]$, $n = 2000$, есть строки разной длины, состоящие из одного символа того же, что и строка-ключ; прочие строки могут иметь префикс, являющийся подстрокой ключа.
23. Максимальный тест: $L = 10000$, $l = 100$, $n = 10000$, у всех строк вхождение 0.
24. Максимальный тест: $L = 10000$, $l = 100$, $n = 10000$, есть строки с вхождением 1.
25. Максимальный тест: $L = 10000$, $l = 100$, $n = 10000$, есть строки с большим числом вхождений.

Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2015 – 2016 учебном году
11 класс

Время выполнения задач — 4 часа

Ограничение по времени — 2 секунды на тест

Ограничение по памяти — 64 мегабайта

11.1. «Вычитаем единицу». Задано трёхзначное натуральное число, не содержащее в своей десятичной записи нулей. Какое число получится, если из каждой десятичной цифры исходного числа вычесть единицу?

Формат входа: Задано единственное трёхзначное натуральное число, большее 200 и не содержащее нулей в своей десятичной записи.

Формат выхода: Выдайте новое число.

Пример

<u>Вход:</u>	<u>Выход:</u>
536	425

11.2. «Выгодное финансовое вложение». Родители Пети Торопыжкин решили открыть банковский вклад на P дней. Выбранный ими банк предлагает несколько видов вкладов, каждый из которых описывается периодом капитализации (в днях) и процентной ставкой, добавляемой к сумме вклада в конце каждого полного срока капитализации. Если вклад снимается до окончания очередного срока капитализации, то за неполный период проценты не начисляются. Помогите Пете написать программу, которая выяснит, который из видов вкладов наиболее выгоден.

Формат входа: В первой строке задано через пробел два целых числа: P — длина в днях периода, на который планируется разместить деньги в банке, и n — количество видов вкладов ($1 \leq P \leq 10^5$, $1 \leq n \leq 100$). В следующих n строках идут описания видов вкладов: два целых числа d_i и s_i — период капитализации в днях очередного вклада и процент по этому вкладу за период капитализации ($1 \leq d_i \leq 10^5$, $0 \leq s_i \leq 20$).

Формат выхода: Выведите единственное целое число — номер наиболее выгодного вида вклада (в том порядке, в каком они задавались на входе; первый вид имеет номер 1).

Пример

<u>Вход:</u>	<u>Выход:</u>
200 2	2
100 20	
9 2	

11.3. «В шеренгу становись!». Класс Пети Торопыжкина строится по росту (по убыванию роста) в шеренгу на урок физкультуры. Некоторые из учеников имеют одинаковый рост, так что их взаимные перестановки не изменяют правильности шеренги, но сделают шеренгу новой. Нужно посчитать количество различных шеренг, в которые может выстроиться Петин класс. Так как это число может быть слишком большим, то посчитайте его по модулю 50 000 (то есть посчитайте остаток от деления этого числа на 50 000).

Формат входа: В первой строке задано n — количество групп школьников, в каждой из которых ребята имеют один рост ($1 \leq n \leq 10^5$). Во второй строке через пробел перечислены величины p_i — численность каждой группы ($1 \leq p_i \leq 10^6$).

Формат выхода: Выдайте единственное целое число — количество различных шеренг, взятое по модулю 10^5 .

Пример

Вход: Выход:

4 15200

1 7 5 4

11.4. «Получить палиндром». Петя Торопыжкин изучает на уроках информатики работу со строками. Он уже изучил первую из операций — обращение подстроки, то есть изменение порядка символов в подстроке, заданной индексом своего первого символа и длиной, на обратный. Его заинтересовало, можно ли при помощи только этой операции превратить заданную непустую строку в *палиндром*, то есть в строку, одинаково читающуюся с начала и с конца. Помогите ему, написав программу, которая для заданной строки будет отвечать на этот вопрос. В случае возможности получения палиндрома программа должна выдавать последовательность операций, приводящую к получению палиндрома, максимально возможного в лексикографическом порядке. Количество команд не должно превосходить длины строки. Если превращение невозможно, программа должна выдать наименьшее количество символов, которые надо удалить из исходной строки, чтобы такое превращение стало возможным.

Формат входа: В первой строке задано единственное натуральное число n , не превосходящее 10^4 — длина строки. В второй строке задана строка, которую хочет обработать Петя. Она состоит из n заглавных латинских символов.

Формат выхода: Если заданную строку можно превратить в палиндром, то в первой строке выдайте YES. Во второй строке выдайте количество операций, за которые может быть получен палиндром, максимальный в лексикографическом порядке (величина, не превосходящая длины строки). В следующих строках описываются операции обращений подстрок (по одной в строке): пара целых чисел через пробел — индекс (отсчитываемый от 1) первого символа обращаемой подстроки и её длина. В случае невозможности превращения строки в палиндром в

первой строке выдайте NO, а во второй выдайте единственное целое число — минимальное количество символов, которые надо удалить из строки, чтобы её всё-таки можно было превратить в палиндром.

Пример 1

<u>Вход:</u>	<u>Выход:</u>
5	YES
XYXZY	2
	3 2
	1 2

Пример 2

<u>Вход:</u>	<u>Выход:</u>
5	NO
YXZXW	2

Примечание: Во втором примере из строки достаточно удалить два символа, например, Y и W, после чего строку можно превратить в палиндром. Удалением одного символа обойтись нельзя.

11.5. «Лягушка-поскакушка». В саду у Пети Торопыжкина много лягушек. Вот и сейчас лягушка скачет по прямой садовой дорожке, вымощенной n квадратными плитками, уложенными в ряд. Вначале она сидит на первой плитке, и, конечно, ей хочется проскакать всю дорожку. Каждый раз лягушка прыгает в направлении нужного ей конца дорожки. Первый прыжок она может сделать максимум на m плиток (с 1-й на $(m+1)$ -ю), а длина каждого следующего прыжка l_i не может превосходить величины $\lfloor k/l_{i-1}^2 \rfloor$, где $\lfloor \cdot \rfloor$ — округление вниз, а l_{i-1} — величина предыдущего прыжка. Петя задумался: за какое минимальное количество прыжков лягушка может выскочить за пределы последней n -й плитки? Напишите программу, которая находит это число.

Формат входа: В первой строке заданы три целых числа n — количество плиток на дорожке, m — максимальная дальность первого прыжка — и k — параметр прыжка ($1 \leq n, m \leq 10^5$, $1 \leq k \leq 1000$).

Формат выхода: Выдайте единственное целое число — минимальное количество прыжков, за которое лягушка может соскочить с дорожки.

Пример

<u>Вход:</u>	<u>Выход:</u>
10 5 10	2

Примечание: В примере лягушка может сначала прыгнуть на одну плитку, а вторым прыжком — на 10 плиток, тем самым оказавшись за пределами дорожки.

Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2015 – 2016 учебном году
11 класс. Разбор решений и идеи тестов

11.1. «Вычитаем единицу». *Задано трёхзначное натуральное число, не содержащее в своей десятичной записи нулей. Какое число получится, если из каждой десятичной цифры исходного числа вычесть единицу?*

Собственно, для решения задачи нужно понять, что ответ есть разность исходного числа и 111, и нет нужды производить разложение числа на цифры и другие сложные операции с данным числом.

Идеи тестов:

1–10. Случайные тесты.

11.2. «Выгодное финансовое вложение». *Родители Пети Торопыжский решили открыть банковский вклад на P дней. Выбранный ими банк предлагает несколько видов вкладов, каждый из которых описывается периодом капитализации (в днях) и процентной ставкой, добавляемой к сумме вклада в конце каждого полного срока капитализации. Если вклад снимается до окончания очередного срока капитализации, то за неполный период проценты не начисляются. Помогите Пете написать программу, которая выяснит, который из видов вкладов наиболее выгоден.*

В целом, решение данной задачи включает две процедуры: подсчет суммарного изменения начальных накоплений при заданном плане и поиск плана, дающего максимальное приращение накоплений. Вторая часть есть стандартный алгоритм поиска максимума.

Первая процедура тоже не является сложной. Пусть r_i — количество полных периодов капитализации i -го вклада: $r_i P \operatorname{div} d_i$. За один период вклад увеличится в $(1 + s_i/100)$ раз: на каждую единицу денег будет начислен процент $s_i/100$. Стало быть, суммарно за r_i периодов капитализации вклад увеличится в $(1 + s_i/100)^{r_i}$ раз. Максимум этих величин по всем i надо найти и выдать номер i , доставляющий этот максимум.

При отсутствии в библиотеке языка специальной функции возведения вещественного числа в степень можно использовать школьную формулу $a^b = e^{b \cdot \ln a}$ ($\ln$ — натуральный логарифм, логарифм по основанию числа e), а функции экспоненты `exp` (возведения числа e в заданную вещественную степень) и `log` логарифма натурального есть во всех языках. Ну или принимая во внимание, что r_i — величина целая, можно возведение в степень заменить повторным произведением.

Идеи тестов:

1. Минимальный тест: $n = 1$.
 2. Максимальный тест: все величины имеют максимальные значения.
- 3–10. Случай

11.3. «В шеренгу становись!». Класс Пети Торопыжскина строится по росту (по убыванию роста) в шеренгу на урок физкультуры. Некоторые из учеников имеют одинаковый рост, так что их взаимные перестановки не изменяют правильности шеренги, но сделают шеренгу новой. Нужно посчитать количество различных шеренг, в которые может выстроиться Петин класс. Так как это число может быть слишком большим, то посчитайте его по модулю 50 000 (то есть посчитайте остаток от деления этого числа на 50 000).

Средняя сложность данной задачи обусловлена необходимостью применения знаний из комбинаторики и соображений относительно работы с числами по заданному модулю.

Известно, что общее количество перестановок из p различных предметов равно $p! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot p$ — факториалу числа p . Эта величина очень быстро растёт с увеличением p ; факториал числа 12 входит в тип `int`, а 13 — уже нет. Именно этим обусловлен то, что в задаче требуется вычислить не общее количество различных шеренг, а остаток от деления этого числа на 50 000.

Если у нас есть две группы из p_1 и p_2 объектов, то с каждой из $p_1!$ перестановок первой группы можно соединить $p_2!$ перестановок второй, что в общем даёт $p_1! \cdot p_2!$ различных шеренг. Вообще, общее количество перестановок в n группах равно произведению $p_1! \cdot p_2! \cdot p_3! \cdot \dots \cdot p_n!$. То есть в задаче требуется вычислить остаток от деления такой величины на 50 000.

Несложно обосновываемый факт из теории чисел, облегчающий эту задачу, заключается в том, что $(a \cdot b) \bmod c = ((a \bmod c) \cdot (b \bmod c)) \bmod c$. То есть, чтобы найти остаток от произведения, можно найти остаток от произведения остатков. Таким образом,

$$p! \bmod c = \left(\dots \left(\left(((1 \bmod c) \cdot (2 \bmod c)) \bmod c \right) \cdot (3 \bmod c) \right) \bmod c \dots \right) \bmod c.$$

То есть при подсчёте факториала вместо умножения на очередной сомножитель уже накопленного произведения (по модулю 50 000) умножаем на остаток от деления этого сомножителя на 50 000 и ищем остаток от деления полученного произведения:

```
res := 1;
for i := 1 to n do
    res := ( res * (i mod 50000) ) mod 50000;
```

Эта идея используется и при вычислении факториалов и при вычислении произведения факториалов.

Некоторая тонкость заключается в том, что факториалы (вернее, их остатки) надо просчитать заранее, так как независимое вычисление факториала для каждого из чисел, в целом, может занять слишком большое время. Поэтому решение данной задачи выглядит следующим образом:

```
const
  MaxP = 1000000;
  K = 50000; var
  f: array[1..MaxP] of integer;
  res, p, n, i: integer;
begin
  (* считаем факториалы *)
  f[1] := 1;
  for i := 2 to MaxP do f[i] := (f[i-1] * (i mod K)) mod K;

  (* считываем числа и считаем результат *)
  res := 1;
  read(n);
  for i := 1 to n do begin
    read(p);
    res := (res * f[p]) mod K;
  end;
  writeln(res);
end.
```

Идеи тестов:

1. Минимальный тест: $n = 1, p_1 = 1$.
2. Минимальный тест: $n = 1, p_1 = 5$.
3. Минимальный тест: $n = 1, p_1 = 50$.
4. $n = 10$, все $p_i = 1$.
5. $n = 10$, все p_i различные, меньше 12.
6. $n = 10$, все p_i различные, в том числе и бóльшие 12.
7. $n = 100$, все $p_i = 1$.
8. $n = 100$, все p_i различные, меньше 12.
9. $n = 100$, все p_i различные, в том числе и бóльшие 12.
10. Максимальный тест: $n = 10^5$, все $p_i = 1$.
11. Максимальный тест: $n = 10^5$, все p_i различные, меньше 12.
12. Максимальный тест: $n = 10^5$, все p_i различные, в том числе и бóльшие 12.
- 13–20. Случайные тесты.

11.4. «Получить палиндром». Петя Торопыжский изучает на уроках информатики работу со строками. Он уже изучил первую из операций — обращение

подстроки, то есть изменение порядка символов в подстроке, заданной индексом своего первого символа и длиной, на обратный. Его заинтересовало, можно ли при помощи только этой операции превратить заданную непустую строку в палиндром, то есть в строку, одинаково читающуюся с начала и с конца. Помогите ему, написав программу, которая для заданной строки будет отвечать на этот вопрос. В случае возможности получения палиндрома программа должна выдавать последовательность операций, приводящую к получению палиндрома, максимально возможного в лексикографическом порядке. Количество команд не должно превосходить длины строки. Если превращение невозможно, программа должна выдать наименьшее количество символов, которые надо удалить из исходной строки, чтобы такое превращение стало возможным.

По мнению программного комитета, данная задача не очень сложна идейно, но требует аккуратного технического исполнения, так что, в целом, сложность этой задачи выше среднего.

Для работающих на Паскале некоторую сложность может вызвать ввод столь длинной строки: стандартный тип `string` хранит лишь строки длиной не более 255 символов, а чтение такой строки в переменную типа `AnsiString` целиком не поддерживается операторами `read/readln`. Однако при наличии длины рабочей строки среди входных данных позволяет считывать её посимвольно и хранить в массиве `array[1..MaxLen] of char`, где `MaxLen` — максимальная длина строки, константа 10^4 .

Дальнейшая обработка строки обуславливается следующим классическим наблюдением: строку можно превратить в палиндром, если количество символов, встречающихся в этой строке нечётное число раз, равно 0 или 1. Соответственно, после того, как строка считана в память (или в процессе считывания, если считывание идёт посимвольно) подсчитываем количество символов каждого типа (от А до Z). Если «нечётных» видов символов больше 1, то строку непосредственно превратить в палиндром нельзя, а чтобы это можно было сделать, нужно удалить из строки по одному символу каждого «нечётного» вида кроме одного.

Если же «нечётных» видов символов 0 или 1, то начинаем превращать строку в палиндром, максимальный в лексикографическом порядке. Для этого ищем символ максимальный из тех, что ещё не стоят на своих местах (для этого нужно помнить, сколько символов уже поставлено) и входящих в необработанную часть строки хотя бы два раза, и обращаем соответствующую подстроку так, чтобы он встал на место, следующее за уже сформированной начальной частью строки. Затем ищем подстроку такую, чтобы такой же символ встал на место перед хвостовой сформированной частью строки. Каждое обращение подстроки выполняется буквально: переставляются соответствующие элементы массивов. По ходу дела, запоминаем проделанные операции. После конца обработки, выдаем в результат их количество и описания. Заметим, что если у нас есть символ, входящий нечётное число раз, то его непарное вхождение автоматически окажется в середине

строки в результате такой обработки.

Время работы алгоритма — $O(n^2)$, где n — длина входной строки. Действительно, нам нужно расставить n символов на свои места, для чего надо переставлять местами символы подстроки, длина которой в худшем случае есть длина всей строки. Заданные ограничения дают возможность сделать описанную обработку в рамках ограничений по времени.

Поскольку в случае возможности превращения строки в палиндром требуемая последовательность операций, вообще говоря, неединственна (например, в первом из примеров к задаче можно поменять местами первую и вторую операции), то проверка задачи ведётся при помощи анализатора, который проделывает следующие действия. Считывается строка из входных данных, ответ жюри и ответ участника. Проверяется совпадение типа ответов (YES/NO). Если они не совпадают, ответ участника признаётся неверным. В случае совпадения ответов типа NO сравниваются количества символов, требуемых к удалению из строки. В случае совпадения ответов типа YES проверяется, что количество операций, выданных участником, не слишком большое. Затем проводятся операции, выданные участником, и на копии строки — операции, предложенные жюри. Результаты сравниваются. В случае несовпадения результатов считается, что участник не получил палиндром, максимальный в лексикографическом порядке, и его ответ отвергается.

Идеи тестов:

1. Строка длины 1.
2. Строка длины 2, разные символы.
3. Строка длины 2, одинаковые символы.
4. Строка длины 3, три вида символов.
5. Строка длины 3, два вида символов, строка ещё не палиндром.
6. Строка длины 3, два вида символов, строка уже палиндром.
7. Строка длины 4, два вида символов, строка ещё не палиндром.
8. Строка длины 4, два вида символов, строка уже палиндром, но не максимальный.
9. Строка длины 4, два вида символов, строка уже максимальный палиндром.
10. Строка длины 4, три вида символов.
11. Строка длины 4, четыре вида символов.
12. Строка длины 100, много видов символов, каждого — чётное число, строка ещё не палиндром.
13. Строка длины 100, много видов символов, каждого — чётное число, строка уже палиндром, но не максимальный.
14. Строка длины 100, много видов символов, каждого — чётное число, строка уже максимальный палиндром.
15. Строка длины 101, много видов символов, каждого, кроме одного — чёт-

ное число, строка еще не палиндром.

16. Строка длины 101, много видов символов, каждого, кроме одного — чётное число, строка уже палиндром, но не максимальный.
17. Строка длины 101, много видов символов, каждого, кроме одного — чётное число, строка уже максимальный палиндром.
18. Строка длины 100, много видов символов, много «нечётных» видов символов.
- 19–25. То же, что в тестах 12–18, только длина строки 1000/1001.
- 26–32. То же, что в тестах 12–18, только длина строки максимальная $10^4/99999$.
- 33–36. Случайные тесты, длина строки 30–50.
- 37–40. Случайные тесты, длина строки 100–200.
- 41–45. Случайные тесты, длина строки 1000–5000.
- 46–50. Случайные тесты, длина строки 90000–100000.

11.5. «Лягушка-поскакушка». *В саду у Пети Торопыжского много лягушек. Вот и сейчас лягушка скачет по прямой садовой дорожке, вымощенной n квадратными плитками, уложенными в ряд. Вначале она сидит на первой плитке, и, конечно, ей хочется проскакать всю дорожку. Каждый раз лягушка прыгает в направлении нужного ей конца дорожки. Первый прыжок она может сделать максимум на t плиток (с 1-й на $(t + 1)$ -ю), а длина каждого следующего прыжка l_i не может превосходить величины $\lfloor k/l_{i-1} \rfloor$, где $\lfloor \cdot \rfloor$ — округление вниз, а l_{i-1} — величина предыдущего прыжка. Петя задумался: за какое минимальное количество прыжков лягушка может выскочить за пределы последней n -й плитки? Напишите программу, которая находит это число.*

Данная задача, по мнению программного комитета, является сложной, так как привлекает не вполне тривиальный вариант принципа динамического программирования. Кроме того, для получения полного решения требуется весьма аккуратная техническая реализация данного принципа.

Для начала заметим, что лягушке нельзя прыгать больше, чем на $L = \lfloor \sqrt{k} \rfloor$ плиток, кроме как в последнем прыжке. В противном случае следующий прыжок можно будет сделать не более, чем на ноль плиток, то есть лягушка после слишком длинного прыжка дальше прыгать не сможет. Максимально допустимая длина разрешённого прыжка (при наибольшем значении k) есть $\lfloor \sqrt{1000} \rfloor = 31$ плитки.

Основной структурой данных будет двумерный массив `plates` целых чисел. В ячейке с индексом $[i, j]$ будет храниться количество прыжков, за которые можно добраться до i -й плитки с величиной последнего прыжка j ; индекс i меняется в диапазоне от 1 до n , индекс j — в диапазоне от 1 до L . В языках без возможности динамического создания массивов следует отвести массив с запасом с диапазоном $1..10000$ по первому индексу и диапазоном $1..31$ по второму. Если ячейки массива сделать типа `int`, то потребный объём памяти будет несколько больше мегабайта. Соответственно, размер этого массива будет определять и наихудшее

время работы программы — $O(n \cdot \sqrt{k} \cdot \sqrt{k}) = O(nk)$: n — диапазон цикла обработки плиток, $\sqrt{k}$ — диапазон цикла при обработке очередной плитки, $\sqrt{k}$ — наибольший диапазон прыжка. От параметра m сложность алгоритма не зависит.

Кроме того, храним переменную **res**, которая хранит текущий минимум прыжков, выводящих лягушку с дорожки.

В начале программы **res** равна $+\infty$, то есть величине, заведомо большей, чем потребное количество прыжков. Например, в качестве этой величины можно выбрать 10^6 . В массиве **plates** все ячейки с индексом $i \geq 2$ также заполняются бесконечностью (эти данные ещё не просчитаны), а ячейки с индексом $i = 1$, заполняются нулями — лягушка уже стоит на первой плитке, она туда «доберётся» за 0 прыжков.

Вначале надо проверить условие $m + 1 > n$. Если это так, то лягушка за один прыжок может соскочить с дорожки. В этом случае программу следует завершить с результатом 1.

В противном случае запускаем обработку массива **plates**. Сначала отдельно обрабатывается первый прыжок; особенность его обработки в том, что у него нет предыдущего прыжка и его дальность ограничена минимумом величин m и L (то есть тем, что можно сделать, и тем, что разумно сделать). Соответственно, для всех s от 1 до $\min\{m, L\}$ присваиваем **plates[s+1, s] := 1**.

Затем обрабатываем последующие прыжки. Пересчёт идёт по индексам i от 2 до n , а для каждого значения индекса i по индексу j от 1 до L . Если значение **plates[i, j]** меньше бесконечности, то есть если на i -ю плитку лягушка может допрыгать с последним прыжком величины j , то смотрим, куда она может прыгнуть дальше. Для всех значений счётчика s от 1 до $\lfloor k/j^2 \rfloor$ (то есть для всех возможных величин нового прыжка) проверяем, если $i + s > n$ (то есть если лягушка спрыгивает с дорожки), то пересчитываем **res**:

$$\mathbf{res} = \min\{\mathbf{res}, \mathbf{plates}[i, j] + 1\}.$$

Если же $i + s \leq n$, то пересчитываем информацию по плитке, куда доскочит лягушка:

$$\mathbf{plates}[i+s, s] = \min\{\mathbf{plates}[i+s, s], \mathbf{plates}[i, j] + 1\}.$$

После окончания работы этого двойного цикла переменная **res** будет содержать ответ.

Примечание: Возможно, задача решается каким-либо «жадным» алгоритмом, однако он не вполне очевиден, и подход, основанный на принципе динамического программирования, достаточно разумен.

Идеи тестов:

1. Минимальный тест: $n = 1, m = 1, k = 1$.
2. Минимальный тест: $n = 10, m = 1, k = 1$.

3. Минимальный тест: $n = 10, m = 10, k = 1$.
4. Минимальный тест: $n = 10, m = n/2 = 5, k = 1$.
5. Минимальный тест: $n = 10, m = 10 - 1 = 9, k = 1$.
6. Минимальный тест: $n = 10, m = 1, k = 10$.
- 7–11. То же, что в тестах 2–6, только вместо 10 — 100.
- 12–16. То же, что в тестах 2–6, только вместо 10 — 10^5 (1000 для k).
17. Максимальный тест: $n = 10^5, m = 10^5, k = 1000$.
18. Максимальный тест: $n = 10^5, m = 10^5 - 1 = 99999, k = 1000$.
- 19–22. Случайные тесты, $m + 1 > n, 10 \leq n \leq 100$.
- 23–26. Случайные тесты, $m + 1 < n, m > L, 10 \leq n \leq 100$.
- 27–30. Случайные тесты, $m + 1 < n, m < L, 10 \leq n \leq 100$.
- 31–42. Такие же случайные тесты, $1000 \leq n \leq 10000$.
- 43–50. Такие же случайные тесты, $90000 \leq n \leq 100000$.