

Задача А. Два знака

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Вам дано трехзначное число, и в вашем распоряжении три арифметических операции: сложение, вычитание и умножение. Вы должны между каждыми двумя цифрами поставить знак одной из операций таким образом, чтобы после вычисления результата выражения получилось максимальное число. Все операции должны быть различными, то есть вы не можете дважды использовать одну и ту же операцию.

Цифры менять местами нельзя. При вычислении результата выражения приоритеты операций сохраняются.

Формат входных данных

Вам дано одно целое положительное число n ($100 \leq n \leq 999$).

Формат выходных данных

Выведите максимальное число, которое может получиться в результате вычисления получившегося выражения.

Пример

стандартный ввод	стандартный вывод
128	17

Замечание

В примере мы расставляем знаки следующим образом: $1 + 2 \cdot 8 = 17$.

Задача В. Аугментация

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Сегодня нашу жизнь невозможно представить без машинного обучения. Вы сталкиваетесь с ним, когда фотографируете красивый закат, играете в новую стрелялку или создаёте аватарку с фильмом Барби в социальных сетях. Всё это результат труда огромного числа учёных и программистов.

Для достижения высокого качества работы алгоритмов машинного обучения требуется большое число размеченных данных. Алгоритмы машинного обучения в некотором смысле «запоминают» то, что им показывают, и потом применяют свои «знания» к новым объектам. Таким образом, зачастую качество модели тем выше, чем больший объём данных она увидела при обучении. Для сложных алгоритмов необходим ещё больший объём размеченных данных. Для некоторых задач можно использовать существующие наборы данных, такие как ImageNet или Google's Open Images. Они открыты для общего использования и постоянно расширяются за счёт энтузиастов. Но для решения большинства задач необходимы специальные данные, получение и разметка которых, вообще говоря, могут дорого стоить или вообще быть недоступными. Например, для разметки программного кода нужен специалист, знающий и понимающий исследуемый материал. Кроме того, человек работает гораздо медленнее машины, поэтому получить необходимую размеченную выборку иногда просто невозможно.

Одним из способов увеличения выборки является **аугментация данных**. Существующий набор размеченных данных искусственно расширяется за счёт незначительных преобразований, не меняющих облик объекта с точки зрения человека. Так для изображений используются всевозможные вращения, отображения, зашумления и искажения. То есть, если вы добавите сотню чёрных маленьких квадратиков на картинку с автомобилем, то другой человек на картинке всё ещё будет видеть автомобиль. Для естественных языков применяются различные эвристики с перестановкой слов в предложениях, предложений в текстах, замены слов на синонимы. Аугментация используется и для звуковых данных: изменение частот и амплитуд, зашумление звуковой дорожки. Такие преобразования, на первый взгляд, очень просты, но в то же время дают результат: качество обученных моделей возрастает.

В этой задаче мы хотим рассказать вам об исследовании, в котором описанный метод применяется для нахождения плагиата среди посылок участников констестов. В этом исследовании исходные данные являются набором программ на некоторых языках программирования.

Код на любом языке программирования — это строго структурированные данные, поэтому изменение порядка операндов у операции или нарушение последовательности инструкций сильно влияют на результат работы программы. Простые методы в этом случае неприменимы. Необходим глубокий анализ, который доступен благодаря возможностям компилятора. Именно он отвечает за всевозможные трансформации программ, сохраняющие семантику. Такие преобразования называются *консервативными*. Они сохраняют семантику исходного кода. В первую очередь, это оптимизации, позволяющие сэкономить память или ускорить выполнение программы. Здесь мы приведём примеры различных консервативных преобразований, которые позволяют многократно увеличить ограниченную выборку.

Исходный код представляет собой последовательность символов, с которой затруднительно работать, поэтому в данной задаче рассматривается более структурированное промежуточное представление программного кода. Это вынужденное ограничение, однако в качестве признаков для машинного обучения такая структура представляет большую ценность. Все преобразования реализуются над промежуточным представлением кода в компиляторной инфраструктуре LLVM, на которой реализован, в том числе, компилятор Clang. LLVM предоставляет функционал для трансформации промежуточного представления, что позволяет реализовывать собственные преобразования.

Для лучшего понимания трансформаций кода введём несколько определений.

- **Базовый блок** (*basic block*, *BB*) — в программировании и теории компиляторов это прямолинейный участок кода, не содержащий в себе ни операций передачи управления, ни точек, на

которые управление передаётся из других частей программы. Имеется лишь два исключения:

- точка, на которую выполняется переход, является первой инструкцией в базовом блоке,
- базовый блок завершается инструкцией перехода.

Базовый блок	Не является базовым блоком
<pre>int a = 0; a += 5; int b = a + 2; a += b;</pre>	<pre>if (a > 10) return 5; else return 15;</pre>

- **Граф потока управления** (*control flow graph, CFG*) — в теории компиляции это множество всех возможных путей исполнения программы, представленное в виде графа. В графе потока управления каждый узел (вершина) графа соответствует *базовому блоку*. Направленные дуги используются в графе для представления инструкций перехода. Также граф потока управления содержит два специализированных блока:

- входной блок, через который управление входит в граф,
- выходной блок, который завершает все пути в данном графе.

- **Доминирующий базовый блок.** Базовый блок *A* доминирует над базовым блоком *B*, если любой путь, ведущий из входного блока в *B*, проходит также через *A*.

Перейдём к трансформациям.

Для ознакомления авторы исследования предлагают четыре вида консервативных трансформаций графа потока управления: изменение порядка инструкций в базовом блоке, перемещение инструкции внутри функции, вынос части графа потока управления в отдельную функцию, преобразование цикла с предусловием в цикл с постусловием. Все эти трансформации меняют вид кода программы, что изменяет входные данные для машинного обучения, при этом семантика трансформированного кода сохраняется. Помимо изучения списка трансформаций в задаче требуется считать натуральное число и прибавить к нему количество языков программирования, упомянутых в условии задачи.

- **Перестановка инструкций в базовом блоке.** Внутри одного базового блока инструкции можно поменять местами, когда они никак не влияют на результат выполнения всего блока, например, можно менять местами порядок объявления переменных, не зависящих друг от друга.
- **Перестановка инструкций внутри функции.** Для перемещения инструкций внутри функции необходимо учитывать зависимость по управлению, поэтому здесь можно переместить инструкцию только так, чтобы она доминировала все зависящие от неё инструкции. Например, изначально в языке C объявлять локальные переменные можно было только в начале функции, в поздних версиях языка и его наследнике C++ от этой идеи отказались, руководствуясь тем, что переменную лучше объявлять как можно ближе к месту её использования. Таким образом, объявление переменной можно перенести из начала функции в место её первого использования.
- **Преобразование цикла с предусловием в цикл с постусловием.** Напомним, что циклом с предусловием является цикл, где для выполнения первой и последующих операций проверяется некоторое условие (циклы `for` и `while` в языке Python), в цикле с постусловием первая итерация выполняется всегда (`repeat-until` в языке Pascal). Таким образом, цикл с предусловием всегда можно преобразовать в цикл с постусловием, если вынести проверку условия перед первой итерацией в условный оператор, а цикл с постусловием поместить в тело этого условного оператора.

- **Вынесение части функции в отдельную.** Вынесение части функции в отдельную — преобразование, существенно меняющее граф потока управления. Для выделения новой функции необходим участок графа потока управления с одним входом и одним выходом.

Организаторы олимпиад предпринимают попытки обнаружить среди участников нарушителей правил, благодаря этому появляются размеченные наборы примеров. Авторы исследования использовали решения контестов студентов первого курса факультета ВМК МГУ. С помощью описанных преобразований им удалось увеличить изначальный набор данных в пять раз. В результате чего качество работы алгоритмов улучшилось на 3-5 процентов, что является существенным успехом в задачах машинного обучения, что доказывает эффективность аугментации.

Формат входных данных

На вход программе подается одно целое число N ($1 \leq N \leq 10^9$).

Формат выходных данных

Выведите ответ на задачу.

Пример

стандартный ввод	стандартный вывод
10	1020437

Замечание

Тестовый пример приведён для демонстрации формата ввода и вывода, ответ на него не является верным.

Задача С. Экскурсия

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

На Всероссийскую командную олимпиаду школьников по программированию приехало n делегаций из разных городов, i -я делегация состоит из a_i человек. Организаторы олимпиады приготовили экскурсию для участников и сопровождающих, но поучаствовать в ней смогут не более k человек. Все гости олимпиады хотят её посетить, но коллективы не планируют разделяться — каждая делегация должна либо поехать на экскурсию в полном составе, либо пропустить её.

Помогите организаторам выбрать максимальное количество делегаций, которые смогут поучаствовать в экскурсии.

Формат входных данных

Первая строка содержит два целых числа n и k ($2 \leq n \leq 1000$, $3 \leq k \leq 10^8$) — количество делегаций и мест на экскурсии.

Вторая строка содержит n целых чисел a_i ($3 \leq a_i \leq 10^5$) — количество человек в i -й делегации.

Формат выходных данных

Выведите одно число — максимальное количество делегаций, которые смогут поучаствовать в экскурсии.

Примеры

стандартный ввод	стандартный вывод
3 7 5 3 4	2
10 300 85 66 72 31 15 27 16 99 32 16	8

Задача D. Ипотека длиной в жизнь

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

От вас требуется решить типовую задачу из ЕГЭ.

Некто Александр недавно переехал в Москву. Однако всего один месяц жизни в общежитии заставил его задуматься о покупке квартиры, поэтому он обратился в Современный Банк Индустриального Развития. В СБИРе ему предложили ипотеку под r процентов годовых.

Александр подсчитал, что он сможет выплачивать по p рублей в год, жить ему осталось n лет, а желаемая квартира стоит s рублей.

Формально процесс выплаты ипотеки можно описать так:

- В январе первого года Александр оформляет ипотеку на s рублей.
- В феврале каждого года остаток по кредиту возрастает на r процентов и округляется с точностью до копеек в пользу банка (в большую сторону).
- С марта по декабрь каждого года заёмщик выплачивает не более p рублей, уменьшая долг на соответствующую сумму.
- В конце декабря последнего года выплат остаток по кредиту должен составлять 0 рублей 0 копеек, что будет означать его закрытие.
- Кредит должен быть погашен не более, чем за n лет.

Сможет ли Александр дожить до дня закрытия ипотеки?

Формат входных данных

В первой строке даны четыре целых числа n, r, p, s ($1 \leq n \leq 200, 1 \leq r \leq 1000, 1 \leq p, s \leq 10^9$) — количество лет, на которые взята ипотека, процентная ставка, размер годовой выплаты и размер ипотеки.

Формат выходных данных

В единственной строке выведите `HeWonThisLife`, если Александр сможет закрыть ипотеку в соответствии с требованиями, и `GGWP` в противном случае.

Примеры

стандартный ввод	стандартный вывод
3 12 8000000 20000000	GGWP
3 13 555 1234	HeWonThisLife

Задача Е. Подарок для Наннали

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Сегодня k -й день рождения будущего короля Британии Лелуша Ламперужа. Однако никакого праздничного настроения у Лелуша нет, молча сидя на диване, он осмысляет свою жизнь.

В детстве с Лелушем и его сестрой Наннали случилось несчастье, и, чтобы сестра не грустила всю оставшуюся жизнь, Лелуш решил подарить ей на день рождения половину Британии. Британией он тогда (как и сейчас) не правил, поэтому приступил к разработке плана по захвату империи, который завершил ровно в свой день рождения, когда ему исполнилось n лет.

На самом деле Британию можно было бы захватить и без завершённого плана, но для этого потребовался бы **Geass**, который Лелуш получил бы в случае успешного побега его подруги С.С. Заполучив **Geass** или разработав план, Лелуш мог приступить к захвату Британии в любой момент (даже в тот же день, когда он завершил план или получил **Geass**). Захват Британии занимает ровно 1 год и должен был бы завершиться точно в день рождения его сестры.

Сидя в одиночестве, Лелуш задумался, сколько же у него было возможностей сделать подарок для Наннали?

Вам требуется решить эту задачу для t Лелушей из параллельных вселенных.

Формат входных данных

Каждый тест содержит несколько наборов входных данных.

В первой строке записано целое число t ($1 \leq t \leq 1000$) — количество наборов входных данных. Далее следуют их описания.

В первой строке набора даны два целых числа n и k ($5 \leq n \leq k \leq 100$) — возраст, когда Лелуш завершил разработку плана по захвату Британии, и текущий его возраст.

Во второй строке набора записаны две строчки через пробел $date_1$ и $date_2$ ($date_1$ хронологически раньше $date_2$) — день рождения Лелуша и дата побега С.С. в формате $yyyy/mm/dd$.

В третьей строке набора записан день рождения Наннали в формате mm/dd .

Гарантируется, что все даты в тестах являются корректными, год лежит в диапазоне $[1, 9999]$ (при необходимости год дополняется лидирующими нулями для соответствия формату), и ни день рождения брата, ни день рождения сестры, ни день побега С.С. не приходятся на 29 февраля.

Формат выходных данных

Для каждого набора входных данных выведите единственное число — количество упущенных возможностей Лелуша сделать подарок Наннали.

Примеры

стандартный ввод	стандартный вывод
2 7 10 1994/04/01 2003/01/01 04/01 7 10 1994/04/01 2000/01/01 02/01	3 4
1 80 100 0001/01/01 0080/01/01 01/01	21

Задача F. Весёлые замены

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Весёлой заменой в строке из маленьких латинских букв называется замена одной гласной буквы на любую другую гласную, гласными буквами в латинском алфавите считаются а, е, i, о, u, y. Остальные буквы в строке при этом не меняются. Например, `lerka` → `lipka` и `lerka` → `lapka` — весёлые замены, а `lerka` → `kerka` и `lerka` → `lapki` — нет.

Вам дана исходная строка и список строк. Для каждой строки из списка проверьте, можно ли получить её из исходной после нескольких (возможно, нуля) весёлых замен.

Формат входных данных

Первая строка содержит исходную строку s ($1 \leq |s| \leq 2000$).

Вторая строка содержит число n ($1 \leq n \leq 2000$) — количество строк в наборе.

Следующие n строк содержат по одной строке t_i из набора ($1 \leq |t_i| \leq 2000$).

Все строки во входных данных состоят только из маленьких латинских букв.

Формат выходных данных

Для каждой строки из списка выведите YES, если её можно получить из исходной после нескольких (возможно, нуля) весёлых замен, и NO в противном случае.

Примеры

стандартный ввод	стандартный вывод
lerka	YES
4	YES
lipka	NO
lapka	YES
kerka	
lapki	
yblsla	NO
4	NO
ybsli	YES
blsl	YES
oblslo	
oblsly	

Задача G. Hokage

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

На плоскости задано n различных прямых вида $y = kx + b$. Назовём множество прямых *хорошим*, если оно не содержит пересекающихся прямых. Требуется разбить все прямые на минимальное количество непересекающихся хороших множеств и вывести это количество.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leq n \leq 10^5$) — количество прямых.

Каждая из следующих n строк содержит два целых числа k, b ($-10^5 \leq k, b \leq 10^5$) — коэффициенты k и b соответствующей прямой.

Гарантируется, что во входных данных нет совпадающих прямых.

Формат выходных данных

Выведите одно целое число — минимальное количество непересекающихся хороших множеств, на которые можно разбить все n прямых.

Примеры

стандартный ввод	стандартный вывод
3 1 -1 2 -2 3 -3	3
4 1 -1 2 -2 1 -3 2 -4	2

Задача Н. Поликарп и mailware

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Поликарп устроился разработчиком в одну крупную компанию. Сотрудники компании пользуются корпоративной почтой, через которую проходят тысячи писем в день. Первой задачей Поликарпа на новой работе стало написание программы, способной определять *потенциально опасные* письма, отправленные злоумышленниками с целью получения секретной информации компании, и передавать их в отдел информационной безопасности для дальнейшего изучения.

Письмо представляет собой текст, состоящий из заглавных и строчных латинских букв, цифр, знаков препинания (точка, запятая, восклицательный и вопросительный знаки), а также пробелов и переводов строк (`\n`). Также у Поликарпа в распоряжении есть список из m опасных слов.

Письмо называется *потенциально опасным*, если оно содержит в тексте каждое из m опасных слов хотя бы по одному разу.

Словом называется непустая непрерывная последовательность символов письма, состоящая из заглавных и строчных латинских букв и цифр, ограниченная слева и справа другими символами и/или началом/концом письма. Например, в письме «I lov3 coding!» подстроки «I», «lov3» и «coding» являются словами, а подстроки «I lov3», «lo», «coding!», «v3 co» и «» — нет.

Каждое из m опасных слов состоит только из строчных латинских букв, поэтому перед тем, как искать слово в списке опасных, необходимо удалить из него все цифры и перевести его в нижний регистр.

Поликарп при устройстве на работу немного приврал в своём резюме, поэтому он без понятия, как решать эту задачу. Помогите ему, написав программу, которая по тексту письма и списку опасных слов определяет, является ли письмо *потенциально опасным*!

Формат входных данных

Первая строка входных данных содержит одно целое число n ($1 \leq n \leq 10^5$) — количество строк в письме.

Далее идёт n строк символов, являющихся текстом письма.

Следующая строка содержит одно целое число m ($1 \leq m \leq 10^5$) — количество опасных слов.

В последующих m строках содержится список опасных слов, по одному слову, состоящему из строчных латинских букв, в каждой строке.

Гарантируется, что все опасные слова в списке различны, а их суммарная длина не превышает 10^5 . Также гарантируется, что длина письма не превышает 10^5 .

Формат выходных данных

Выведите YES, если письмо считается *потенциально опасным*, иначе выведите NO.

Примеры

стандартный ввод	стандартный вывод
3 Cli4ck 0n that l1nk to receive 0n3 million dollars!!! 5 click receive n million dollars	YES
1 I LOVE CODING 2 love chilling	NO

Замечание

В первом примере все опасные слова встречаются в тексте письма хотя бы один раз. Опасное слово «n», например, встречается в тексте даже два раза (оно получается после удаления цифр из слов «0n» и «0n3»), однако количество вхождений не имеет значения, важен только сам факт вхождения опасного слова в письмо.

Во втором примере опасного слова «chilling» нет в тексте письма.

Задача I. ALL IN

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Феликс разбирал свои вещи и обнаружил интересную находку: игральные карты с написанными на них целыми числами (числа написаны ровно с одной стороны каждой карты). Так как Феликс был еще маленьким, и у него было много времени, он придумал игру с этими картами. Он сложил n карт в одну колоду. За один ход он может:

- Выложить любую карту с руки на игровое поле. Выложенные карты кладутся слева направо в том порядке, в каком они были выложены, и остаются лежать там до конца игры. Каждую карту можно выложить не более одного раза.
- Взять в руку верхнюю карту из колоды.

Феликс еще растёт, поэтому его интересуют строго возрастающие последовательности, и он хочет, чтобы в конце игры на игровом поле лежали карты, числа на которых образуют строго возрастающую последовательность длины k . Помогите ему определить минимальное количество ходов, которые он должен сделать, чтобы в результате выложить на игровом поле такую последовательность карт, или скажите, что это невозможно.

У Феликса очень хорошая память, поэтому он запомнил числа на всех картах в порядке, в котором карты лежат в колоде, и готов поделиться этой информацией с вами, чтобы вы ему помогли!

Строго возрастающая последовательность длины k — это последовательность чисел $b_1, \dots, b_k$ такая, что $b_1 < b_2 < \dots < b_k$.

Формат входных данных

Первая строка входных данных содержит два целых числа n и k ($1 \leq n, k \leq 10^5$) — количество карт в колоде и необходимую длину строго возрастающей последовательности.

Вторая строка содержит n натуральных чисел $a_1, a_2, \dots, a_n$ ($-10^9 \leq a_i \leq 10^9$) — числа, написанные на картах в колоде в порядке, начиная с верха колоды (т.е. a_1 — число написанное на верхней карте, a_2 — число, написанное на карте под ней, ... a_n — число, написанное на самой нижней карте).

Формат выходных данных

Выведите одно целое число — минимальное количество ходов, чтобы выложить нужную последовательность, или -1 , если это невозможно.

Примеры

стандартный ввод	стандартный вывод
5 3 7 1 4 2 3	6
3 2 1 1 1	-1

Замечание

В первом примере Феликс может ходить следующим образом:

1. Взять верхнюю карту из колоды, его рука $\{7\}$.
2. Взять верхнюю карту из колоды, его рука $\{7, 1\}$.
3. Выложить карту с числом 1, его рука $\{7\}$, на игровом поле лежит $\{1\}$.
4. Взять верхнюю карту из колоды, его рука $\{7, 4\}$, на игровом поле лежит $\{1\}$.

5. Выложить карту с числом 4, его рука $\{7\}$, на игровом поле лежит $\{1, 4\}$.
6. Выложить карту с числом 7, его рука стала пустой, на игровом поле лежит $\{1, 4, 7\}$ — строго возрастающая последовательность длины 3.

Задача J. Everything, Black and White, No Grey

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Дана строка s , состоящая только из символов «0», «1» (без кавычек, коды символов 48 и 49 соответственно). Разрешается выполнять следующую операцию: поменять местами любые два соседних символа строки s . Нужно минимальным количеством таких операций получить из s строку, в которой нет подстроки «01» (без кавычек) и вывести это минимальное количество операций.

Строка a является подстрокой b , если a может быть получена из b удалением нескольких (возможно, ни одного или всех) символов из начала и нескольких (возможно, ни одного или всех) символов из конца.

Формат входных данных

Первая строка входных данных содержит одно целое число n ($1 \leq n \leq 2 \cdot 10^5$) — длину строки s .

Вторая строка входных данных содержит строку s , состоящую из n символов «0» или «1».

Формат выходных данных

В единственной строке выведите одно целое число — минимальное количество операций, чтобы из s получить строку, в которой нет подстроки «01».

Примеры

стандартный ввод	стандартный вывод
5 01010	3
7 0000000	0

Задача К. Пилот

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

У вас есть сетевой фильтр на n розеток и m устройств с номерами от 1 до m , подключённых к нему. i -й прибор подключён в розетку a_i . Вы хотите подключить ещё одно устройство с номером $m + 1$ к пилоту. Понятно, что мы бы не дали задачу, где нужно проверить, что m меньше n , поэтому допустим, что размеры вилок не позволяют подключать устройства в соседние розетки. Более формально, в любой момент времени, если какое-то устройство подключено в розетку i , не должно быть устройств, подключённых к $(i - 1)$ -й и $(i + 1)$ -й розетке, если эти розетки существуют.

Вы можете отключать произвольное устройство от удлинителя и подключать отключенное ранее устройство обратно в произвольную розетку. Требуется определить, возможно ли с помощью данных операций включить в сеть устройство $m + 1$, оставив все остальные приборы подключёнными, и если да, то определить последовательность действий минимальной длины, которую нужно выполнить, чтобы добиться этого.

Формат входных данных

В первой строке заданы два целых числа n и m ($1 \leq n \leq 10^6$, $0 \leq m \leq n$) — количество розеток на сетевом фильтре и количество устройств, изначально подключённых к нему.

Если $m > 0$, то во второй строке заданы m чисел a_i ($1 \leq a_i \leq n$) — номера розеток, в которые подключены устройства.

Гарантируется, что все a_i различны и что для каждого i от 1 до n в a содержится не более одного числа из множества $\{i - 1, i, i + 1\}$.

Формат выходных данных

В первой строке выведите число k — количество операций в оптимальной последовательности действий (или 0, если подключить новое устройство невозможно).

В следующих k строках выведите саму последовательность действий. Каждое действие должно быть одним из двух:

- «- x » — отключить устройство x от сетевого фильтра ($1 \leq x \leq m + 1$). Перед этим действием устройство x обязательно должно быть подключено к фильтру.
- «+ x y » — подключить устройство x в розетку y ($1 \leq x \leq m + 1$, $1 \leq y \leq n$). Перед этим действием устройство x обязательно должно быть отключено от фильтра, а в розетках с номерами $\{y - 1, y, y + 1\}$ (в случае их существования) не должно быть включённых приборов.

Примеры

стандартный ввод	стандартный вывод
3 1 2	3 - 1 + 1 1 + 2 3
4 2 3 1	0
5 2 1 4	3 - 2 + 2 3 + 3 5

Задача L. Ягами Лайт и страницы из Тетради смерти

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2.5 секунд
Ограничение по памяти:	256 мегабайт

L (Эл) снова начал подозревать Ягами Лайта в том, что тот является Кирой — недавно нашедшим киллером, который карает только преступников. Конечно же Лайт не хочет, чтобы его раскрыли, а потому он решил запутать L'а и находящуюся в его подчинении японскую полицию.

Лайт знает, что у L'а есть доступ к архивам всех тюрем — он располагает списком из n уже умерших от сердечного приступа людей (сердечный приступ — фирменный знак Киры). Конечно же в этот список жертв добавляли в хронологическом порядке. Точно в такой же очередности Лайт записывал имена преступников в «Тетрадь смерти».

Первое правило «Тетради Смерти» звучит так: человек, чьё имя будет записано в эту тетрадь, умрёт. Поэтому Лайт мог писать имена на любой странице. Единственное правило, которого он придерживался — записывать имена на одной странице в одну колонку сверху вниз. То есть если на одной странице оказались имена людей x и y , причём y записан ниже x , то это значит, что y был внесён в список жертв позже x .

Сегодня утром Лайт решил сделать q фальшивых страниц «Тетради смерти», которые никак не могли бы быть настоящими, чтобы передать их L'у и запутать полицию. Утром он написал часть листов и хотел дописать оставшиеся вечером, однако днём L вместе с заместителем начальника японской полиции установил скрытые камеры и микрофоны в комнате Лайта. Единственным выходом было вырвать оставшуюся часть страниц из настоящей «Тетради смерти».

На следующий день L и полиция получили q страниц, однако не смогли проверить какие являются настоящими, а какие — фальшивыми. Помогите L в его расследовании!

Формат входных данных

В первой строке входных данных содержится целое число n ($1 \leq n \leq 2 \cdot 10^5$) — количество имён в списке L.

Во второй строке содержатся n целых чисел a_i ($1 \leq a_i \leq 2 \cdot 10^5$) — имена людей из списка. Так как в мире существует множество людей с одинаковыми именами, то **не** гарантируется, что все имена различны, но одинаковые числа соответствуют одинаковым именам, а разные — разным.

В третьей строке содержится целое число q ($1 \leq q \leq 2 \cdot 10^5$) — количество страниц, которые принёс Лайт.

На следующих $2q$ строках идёт описание страниц. В первой строке описания задаётся целое число m ($1 \leq m \leq 2 \cdot 10^5$) — количество имён на странице. Во второй строке записаны m целых чисел b_i ($1 \leq b_i \leq 2 \cdot 10^5$) — имена людей на этой странице по порядку сверху вниз.

Гарантируется, что сумма количеств имён на всех страницах Лайта не превосходит $2 \cdot 10^6$.

Формат выходных данных

Для каждой страницы выведите YES, если она потенциально может быть настоящей страницей из «Тетради смерти», иначе выведите NO.

Примеры

стандартный ввод	стандартный вывод
7 1 2 3 4 5 6 7 4 4 2 3 5 7 4 1 3 3 4 3 2 5 8 2 1 7	Yes No No Yes
3 1 2 1 4 2 1 1 2 1 2 2 2 1 2 2 2	Yes Yes Yes No

Замечание

Пояснение ко второму примеру:

Первая страница $[1, 1]$ могла быть составлена из листа с 1-м и 3-м элементами.

Вторая страница $[1, 2]$ могла быть составлена из листа с 1-м и 2-м элементами.

Третья страница $[2, 1]$ могла быть составлена из листа с 2-м и 3-м элементами.

Четвёртая страница не могла быть составлена никак, потому что не существует ни одной «Тетради смерти» со страницей $[2, 2]$, которая бы удовлетворяла списку L .

Задача М. Черная металлургия

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	256 мегабайт

В Челябинске есть n заводов. Изначально i -й производит a_i деталей в секунду и работает ровно b_i секунд в год. Таким образом, с i -го завода выходит $a_i \cdot b_i$ деталей в год.

Правительство только отлаживает систему и каждый год вносит q изменений одного из двух типов:

1. Все заводы с номерами от L до R должны увеличить производство на x деталей в секунду. То есть, если $L \leq i \leq R$, то после этого изменения i -й завод начнет выпускать $a_i + x$ деталей в секунду.
2. Все заводы с номерами от L до R должны работать на t секунд в год больше. То есть, если $L \leq i \leq R$, то после этого изменения i -й завод начнет работать $b_i + t$ секунд в год.

После i -го изменения правительство хочет знать, сколько деталей будет произведено за год заводами с номерами от A до B . Так как это число может быть очень большим, выводите его остаток от деления на 998 244 353.

Обратите внимание, что A и B могут различаться для разных запросов. Для лучшего понимания прочитайте формат входных данных.

Формат входных данных

В первой строке находится одно целое число n ($1 \leq n \leq 10^5$) — количество заводов.

В i -й из следующих n строк содержатся по два целых числа a_i и b_i ($1 \leq a_i \leq 10^9$, $1 \leq b_i \leq 3 \cdot 10^7$) — количество деталей, которые i -й завод выпускает за секунду, и количество секунд, сколько i -й завод работает в течение года.

В следующей строке находится одно целое число q ($1 \leq q \leq 10^5$) — количество изменений.

В каждой из следующих q строк сначала содержится целое число $type$ ($1 \leq type \leq 2$), обозначающее тип текущего изменения.

Если $type = 1$, то следом на этой строке идут ещё 5 целых чисел L, R, x, A, B ($1 \leq L \leq R \leq n$, $0 \leq x \leq 10^9$, $1 \leq A \leq B \leq n$), описывающих операцию 1 типа.

Если $type = 2$, то следом на этой строке идут ещё 5 целых чисел L, R, t, A, B ($1 \leq L \leq R \leq n$, $0 \leq t \leq 3 \cdot 10^7$, $1 \leq A \leq B \leq n$), описывающих операцию 2 типа.

Гарантируется, что после всех изменений b_i не будут превосходить $3 \cdot 10^7$.

Формат выходных данных

Для каждого запроса выведите суммарное количество деталей, которые за год произведут заводы с номерами от A_i до B_i , по модулю 998 244 353 в отдельной строке.

Пример

стандартный ввод	стандартный вывод
4	21
1 2	25
3 2	16
1 1	
3 3	
3	
1 2 3 1 1 4	
2 4 4 2 2 4	
1 1 2 3 2 3	

Задача N. Игра в солдатики

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

Лиза в своё свободное время любит играть в солдатики. Она построила n солдатиков в ряд. Рост i -го солдата составляет a_i сантиметров. Она хочет объявить своей подруге Розе игрушечную войну, для этого ей нужно выбрать несколько отрядов солдатиков. Отряд солдатиков должен состоять из подряд идущих солдатиков (какого-то подотрезка), и рост солдатиков в этом подотрезке должен строго возрастать (так отряд будет казаться более запугивающим). Отряд не должен быть слишком маленьким, поэтому количество солдатиков в каждом отряде должно быть не менее k .

Лиза понимает, что, возможно, в её ряду солдатиков нельзя выбрать таким образом отряды. Поэтому она хочет убрать каких-то солдатиков из ряда так, чтобы все оставшиеся солдатики разбились на отряды. Так как она хочет, чтобы в её армии было как можно больше солдатиков, она хочет оставить в ряду максимально возможное количество солдатиков. Помогите ей выкинуть наименьшее количество солдатиков так, чтобы все оставшиеся бились на отрезки длины хотя бы k с строго возрастающим ростом солдатиков, и скажите ей, каких солдатиков тогда нужно убрать.

Заметим, что ряд из 0 солдатиков можно корректно разбить на отряды, поэтому ответ всегда существует.

Формат входных данных

Первая строка входных данных содержит два целых числа n и k ($1 \leq k \leq n \leq 4000$) — количество солдатиков у Лизы и минимальное количество солдатиков в отряде.

Вторая строка содержит n целых чисел a_i ($1 \leq a_i \leq n$) — рост солдатиков в ряду слева направо.

Формат выходных данных

В первой строке выведите одно число m — минимальное количество солдатиков, которых Лизе нужно убрать.

Во второй строке выведите m чисел b_j — номера солдатиков, которых нужно убрать.

Если способов убрать лишних солдатиков несколько при одинаковом количестве, можете вывести любой способ.

Примеры

стандартный ввод	стандартный вывод
5 3 1 2 1 2 3	2 1 2
4 4 1 1 1 1	4 1 2 3 4

Замечание

В первом примере можно также удалить солдатиков с номерами $\{1, 2\}$ или $\{2, 3\}$.

Задача О. Автодополнение

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Однажды Кен решил написать письмо своей Барби о том, как сильно он хочет иметь Моджо-Доджо-Касса-Хаус. Так как бумажные письма давно вышли из моды, а Кен не может отставать от трендов, он решил отправить сообщение с телефона.

На экране телефона есть поле с набираемым словом, а также обычная английская клавиатура с клавишами символов от 'a' до 'z' и клавишей **backspace**. Клавиатура работает так: изначально набранное слово пусто. Действием пользователя считается нажатие на любую клавишу. Нажатие на букву добавляет её к текущему слову, нажатие на **backspace** удаляет последнюю букву текущего слова, если оно не пусто, иначе ничего не делает (но всё равно считается действием).

Также телефон оснащён *системой автодополнения*. Она работает так:

1. Телефон может показывать не более S предложений для автодополнения, каждое можно использовать в качестве действия: оно заменит текущее набранное слово на предлагаемое.
2. В телефоне есть список слов автодополнения, он называется L . Элементы в нём отсортированы в порядке убывания приоритета, т.е. при выборе между двумя элементами L мы возьмём тот, у которого индекс будет меньше.
3. Если Кен в данный момент набрал какую-то строку X , в качестве предложений для автодополнения будут показаны только слова из L , содержащие X в качестве нетривиального префикса. Напомним, что слово A является нетривиальным префиксом B , если B получается из A дописыванием в конец A некоторого **ненулевого** количества символов. В частности, если сейчас Кен набрал некоторое слово X из списка L , X не будет предложен как вариант автодополнения.
4. Если последним действием было нажатие на букву или автодополнение, телефон покажет S слов из L с наибольшим приоритетом, подходящих в качестве вариантов автодополнения. Если таких слов меньше, то и вариантов будет меньше.
5. Если же последним действием было нажатие на **backspace**, предложения формируются в два этапа:
 - (a) Сначала телефон выдаёт S слов из L с наибольшим приоритетом, которые можно предложить сейчас, но нельзя было до последнего действия (например, если было набрано слово 'dog', и Кен нажал клавишу **backspace**, то телефон выдаст предложения, для которых 'do' является нетривиальным префиксом, но 'dog' не является, в том числе может быть показано и само слово 'dog').
 - (b) Если на предыдущем этапе сформировалось меньше S предложений, в качестве оставшихся телефон предложит также те, которые подходили и до нажатия **backspace** (например, если в предыдущем примере $S = 5$ и на предыдущем этапе было найдено всего 3, то телефон добавит ещё два лучших слова, которые содержат 'dog' как нетривиальный префикс).
 - (c) Если после этого предложений оказалось меньше S , то телефон покажет только их.

Итак, вернемся к проблемам Кена. Сейчас он хотел бы набрать только одно слово w из своего сообщения, но перед этим решил развлечься и посчитать количество способов по модулю $10^9 + 7$, которыми он может сделать это ровно за A секунд (за одну секунду Кен нажимает на любую клавишу или же выбирает вариант из автодополнения). Но так как Кен невероятно устал ~~быть~~ ~~всегда вторым~~, он поручил эту задачу вам.

Формат входных данных

В первой строке написана одна строка w , состоящая только из латинских букв в нижнем регистре — слово, которое Кену нужно набрать на телефоне ($1 \leq |w| \leq 50$).

Во второй строке записано целое число n — длина списка L ($0 \leq n \leq 50$).

В следующих n строках в порядке уменьшения приоритета записаны элементы списка L — слова, длиной от 1 до 50 символов, состоящие из букв латинского алфавита в нижнем регистре. Гарантируется, что все слова в списке различны.

В последней строке написано через пробел два целых числа S и A — ограничение на количество предложений автодополнения и количество секунд, которое есть у Кена ($1 \leq S, A \leq 50$).

Формат выходных данных

Выведите единственное число — количество способов набрать слово w ровно за A секунд по модулю $10^9 + 7$.

Примеры

стандартный ввод	стандартный вывод
doesntseemtomatterwhatido 9 doesnt doesnot doesntseem doesntsaw doesntseemto doesntseemfrom doesntseemtomatter doesntseemtodoesntmatter doesntseemtomatterwhat 2 8	3
iamalwaysnumbertwo 2 iamalwaysnumberone iamalwaysnumbertwo 1 6	3
nooneknowshowharditried 3 no noone nooneknow 3 24	998197822

Замечание

В первом тесте есть три варианта:

- $\emptyset \rightarrow \text{doesnt} \rightarrow \text{doesntseem} \rightarrow \text{doesntseemto} \rightarrow \text{doesntseemtomatter} \rightarrow \text{doesntseemtomatterwhat} \rightarrow \text{doesntseemtomatterwhati} \rightarrow \text{doesntseemtomatterwhatid} \rightarrow \text{doesntseemtomatterwhatido}$.
- $\emptyset \rightarrow \text{doesnt} \rightarrow \text{doesntseem} \rightarrow \text{doesntseemt} \rightarrow \text{doesntseemtomatter} \rightarrow \text{doesntseemtomatterwhat} \rightarrow \text{doesntseemtomatterwhati} \rightarrow \text{doesntseemtomatterwhatid} \rightarrow \text{doesntseemtomatterwhatido}$.
- $\emptyset \rightarrow \text{doesnt} \rightarrow \text{doesntseem} \rightarrow \text{doesntseemto} \rightarrow \text{doesntseemt} \rightarrow \text{doesntseemtomatterwhat} \rightarrow \text{doesntseemtomatterwhati} \rightarrow \text{doesntseemtomatterwhatid} \rightarrow \text{doesntseemtomatterwhatido}$.