

Задача А. Документ

Имя входного файла:	<code>document.in</code>
Имя выходного файла:	<code>document.out</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт
Отображение результатов:	полное

Для обеспечения проведения важного государственного мероприятия необходимо выпустить основополагающий документ. Для того, чтобы документ вступил в законную силу, его должны подписать N должностных лиц, причём, порядок, в котором необходимо подписывать документ, фиксирован (для этой задачи будем считать, что должностные лица пронумерованы в этом порядке, начиная от лица номер 1, и заканчивая лицом номер N).

Известно, что у каждого должностного лица на неделе есть фиксированные приёмные дни. Как известно, в неделе семь дней, но должностные лица могут принимать только в рабочие дни (с понедельника по пятницу). У каждого должностного лица есть хотя бы один приёмный день.

Подписание документа одним должностным лицом занимает ровно один день (то есть, если в какой-то день должностное лицо рассматривает документ, то другое должностное лицо уже не может начать рассмотрение документа в этот день).

Утром некоторого понедельника (назовём его первым днём) вам поступил на руки проект документа. Вам требуется написать программу, рассчитывающую минимальное количество дней, необходимое для подписания документа в указанном порядке (если документ не удалось подписать за одну неделю, то вы можете продолжать его подписывать на следующей неделе и т. д.).

Гарантируется, что документ подготовлен настолько хорошо, что у должностных лиц не будет к нему претензий, то есть, каждое должностное лицо утверждает документ ровно за один день с первого раза.

Формат входных данных

В первой строке входного файла задано число N — количество должностных лиц ($1 \leq N \leq 50$). Далее следуют N строк, в каждой из которых через пробел записаны пять целых чисел, соответствующих дням недели с понедельника по пятницу включительно. Число в i -й строке, соответствующее j -му дню недели, равно 0, если i -е должностное лицо не принимает в j -й день недели, иначе число равно 1.

Формат выходных данных

Выведите в выходной файл одно целое число — минимальное количество дней, требуемое для подписания документа.

Примеры

<code>document.in</code>	<code>document.out</code>
2 1 0 1 0 1 1 0 0 1 0	4
2 0 1 0 0 1 1 1 0 0 0	8

Пояснения к примерам

В первом примере документ должны подписать два должностных лица — первое принимает по понедельникам, средам и пятницам, а второе — по понедельникам и четвергам. Наиболее быстрый способ подписать документ таков: вы относите документ первому должностному лицу в понедельник или среду, а второму должностному лицу — в четверг, таким образом, на подписание документа уйдёт 4 дня. Во втором примере первое должностное лицо принимает по вторникам и пятницам, а второе — по понедельникам и вторникам. Даже если вы отнесёте документ первому должностному лицу во вторник, то вы уже не успеете подписать его на первой неделе, вы сможете это сделать только в следующий понедельник, и на весь процесс подписания уйдёт 8 дней.

Задача В. Фродо и монстр

Имя входного файла:	monster.in
Имя выходного файла:	monster.out
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт
Отображение результатов:	нет

Фродо борется с монстром, головы которого последовательно пронумерованы от 1 до K . Монстр — носитель кибернетического генома, и он характеризуется весьма необычными свойствами. Если Фродо отрежет голову с нечётным номером X , тогда у монстра на этом месте вырастет Y голов, где Y — максимальное простое число, меньшее X . При отрезании первой головы новая голова не вырастает. Известно, что на шее монстра не вмещается более 30 000 000 голов (то есть, если сумма Y и оставшихся голов больше 30 000 000, тогда количество голов будет в точности 30 000 000).

Если Фродо отрежет голову с чётным номером Z , тогда у монстра отваливаются все те головы, номера которых в двоичной системе содержат столько единиц, сколько содержит Z . Если монстр лишается всех голов, то бой прекращается.

По результатам каждого удара Фродо (как после вырастания новых голов, так и после того, как некоторые головы отвалились) головы монстра перенумеровываются заново, начиная с 1.

У Фродо нет времени выяснять чётность-нечётность голов монстра, и он режет их без систематизации. К счастью, он имеет лазерную саблю новейшей технологии, которая фиксирует номера отрезаемых голов в момент их отрезания. Вечером поединок прекращается, но усталый Фродо не может считать количество оставшихся голов.

Вам требуется написать программу, которая бы подсчитала количество голов после прекращения поединка.

Формат входных данных

В первой строке входного файла находятся два разделённых пробелом целых числа K ($2 \leq K \leq 30\,000\,000$) и N ($2 \leq N \leq 200\,000$), где K — начальное количество голов монстра, а N — количество отрезаний. В каждой из следующих N строк записано целое число. Эти числа описывают последовательность номеров отрезанных голов. Номера голов корректны, то есть они никогда не превышают общее количество голов монстра в момент удара.

Формат выходных данных

Выведите в выходной файл количество голов на шее монстра после прекращения поединка.

Пример

monster.in	monster.out
7 4 5 9 6 4	5

Пояснение к примеру

После отрезания головы номер 5 у монстра вырастут ещё 3 головы и их станет 9. После отрезания девятой головы у монстра вырастут ещё 7 голов и их станет 15. После отрезания шестой головы у монстра отвалятся головы с номерами 3, 5, 9, 10 и 12, а после отрезания головы номер 4 — головы 8, 2 и 1. В итоге у монстра останется 5 голов.

Задача С. Кэш

Имя входного файла:	cache.in
Имя выходного файла:	cache.out
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт
Отображение результатов:	частичное

Рассмотрим модель некоторой системы, в которой используется набор из N объектов разного размера. Объект может использоваться системой, только если он находится в кэше. Если объекта в кэше нет, то непосредственно перед использованием он должен быть туда помещен (при этом из кэша могут быть удалены другие объекты, чтобы освободить место). Для помещения объекта размера S_i в кэш достаточно, чтобы в нём было суммарно как минимум S_i единиц свободного места. У каждого объекта своя стоимость помещения его в кэш. Стоимость удаления любого объекта из кэша равна нулю. Кэш может содержать объекты суммарным размером не больше C . Вы знаете последовательность, в которой объекты будут использованы системой. Определите какие объекты и когда следует удалять из кэша, чтобы суммарная стоимость помещений объектов в кэш была минимально возможной. Изначально кэш пустой. Объект не может быть помещен в кэш до того как будет запрошен.

Формат входных данных

Первая строка входного файла содержит три целых числа N , C и K ($1 \leq N \leq 18$, $1 \leq C \leq 10^9$, $1 \leq K \leq 100$), где K — количество запросов на помещение объекта в кэш. Вторая строка содержит N целых чисел S_i , где S_i — размер объекта номер i в пределах от 1 до C . Третья строка также содержит N целых чисел: i -е число — стоимость помещения объекта номер i в кэш в пределах от 0 до 10^6 . Четвертая строка содержит K целых чисел в пределах от 1 до N — номера объектов в порядке их использования. Числа в строках разделены пробелами.

Формат выходных данных

Первая строка выходного файла должна содержать одно целое число — минимальную суммарную стоимость помещений объектов в кэш. Затем выведите K строк. i -я строка должна содержать список объектов удаляемых из кэша перед использованием i -го в списке объекта. Первое число в строке — количество удаляемых объектов K_i , затем K_i чисел — номера удаляемых объектов. Числа в строках необходимо разделять пробелами.

Примеры

cache.in	cache.out
2 10 3	5
9 8	0
2 1	1 1
1 2 1	1 2
2 10 3	3
1 3	0
2 1	0
1 2 1	0

Задача D. T9

Имя входного файла:	t9.in
Имя выходного файла:	t9.out
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт
Отображение результатов:	частичное

Многие пользователи мобильных телефонов при наборе sms-сообщений используют режим T9. При этом сообщения, например на английском языке, они набирают следующим образом.

Для набора слова по буквам соответствующая букве кнопка нажимается один раз, вне зависимости от того, сколько букв соответствуют этой кнопке, и какой по счету идет нужная буква (см. картинку), а программа в телефоне подбирает из имеющегося словаря подходящее для данной комбинации кнопок слово. Если подходящих слов несколько, то в первую очередь предлагается наиболее часто встречающееся слово (изначально слова одинаковой встречаемости предлагаются по алфавиту).

1 .	2 abc	3 def
4 ghi	5 jkl	6 mno
7 pqrs	8 tuv	9 wxyz

Если слово не подошло, то пользователь нажимает кнопку ‘*’ и программа предлагает второе по встречаемости слово, образуемое той же комбинацией кнопок (в первую очередь следующее слово с той же частотой встречаемости, если такое имеется). Если и оно не подошло, то кнопка ‘*’ нажимается еще раз и т.д. Для простоты будем считать, что современные модели телефонов содержат полный словарь используемых слов, и нужное слово обязательно найдется. Когда предлагаемое слово подошло, пользователь нажимает на кнопку “пробел”, нажимает на кнопку ‘1’ (последняя соответствует набору знака препинания), или заканчивает набирать сообщение. Когда знак препинания не подошёл, опять же нажимается кнопка ‘*’, до тех пор пока не появится требуемый знак. После набора пробела или знака препинания пользователь может ввести ещё один пробел или знак препинания, начать набирать следующее слово или закончить набирать сообщение. Будем считать, что пользователю достаточно трех знаков препинания, и они предлагаются в следующем порядке: точка (‘.’), запятая (‘,’), вопросительный знак (‘?’).

После того как пользователь “утвердил” набранное слово (нажав на пробел или ‘1’), частота его встречаемости в словаре увеличивается на 1, и новое значение частоты учитывается, в том числе, и при наборе в режиме T9 остальных слов того же сообщения. При этом данное слово будет предлагаться первым среди слов с такой же частотой встречаемости, порядок предложения остальных слов остается неизменным. Когда появится еще одно слово с той же частотой, то уже оно будет предлагаться первым, не меняя порядка остальных, и т. д.

Вам требуется написать программу, которая по имеющемуся словарю, содержащему первоначальные характеристики частоты встречаемости того или иного английского слова, и известной последовательности нажатий кнопок пользователем при наборе sms-сообщения в режиме T9 воспроизведет появившееся на экране сообщение.

Формат входных данных

В первой строке входного файла находится целое число N ($3 \leq N \leq 50\,000$) — количество слов в словаре. В каждой из следующих N строк записаны одно слово словаря и через пробел натуральное число F ($1 \leq F \leq 1000$) — первоначальное значение частоты встречаемости этого слова (чем больше значение, тем чаще встречается данное слово). Числовая характеристика частоты встречаемости отделена от слова ровно одним пробелом. Слова в словаре состоят только из строчных английских букв и расположены в алфавитном порядке. Длина слова не превышает 20 символов. Все слова не пустые и различные.

Последняя строка файла состоит из цифр от 1 до 9 и символов “пробел” и ‘*’, обозначающих последовательность нажатий кнопок при наборе сообщения. Длина этой строки не превосходит 100 000 символов.

Формат выходных данных

Выведите в выходной файл текст sms-сообщения.

Примеры

t9.in	t9.out
5 ad 2 be 1 not 10 or 5 to 50 86 23* 67 668 86 231**	to be or not to be?
3 act 1 bat 1 cat 1 228* 228** 228** 228**1	bat cat act bat.

Пояснения к примерам

Во втором примере сначала программой будет предложено слово “act”, затем “bat”, которое будет принято пользователем. После чего частота встречаемости слова “bat” станет равной 2. Поэтому при повторном запросе той же числовой комбинации сначала предложено слово “bat”, затем “act” и “cat”. В третий раз слова будут предлагаться в порядке “cat” (как новое слово с частотой 2), “bat” (частота 2), “act” (и его частота сравнивается с частотой остальных слов), и в последнем случае слова будут предлагаться в порядке “act”, “cat”, “bat”.