Задача A. Казино

Имя входного файла:	`blackjack.in`
Имя выходного файла:	`blackjack.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Евгений — владелец небольшого казино, и в этом казино есть только одна игра. В игре используется бесконечная колода карт, то есть для каждого типа карты известна вероятность появления такой карты следующей в колоде, и вероятности не меняются со временем. Каждый тип карты имеет своё достоинство, выраженное целым числом.

Игра происходит следующим образом. Вначале крупье берёт по одной карте из колоды. Как только сумма достоинств взятых им карт превысит число S, он отдаёт последнюю взятую им карту игроку и перестаёт брать новые карты. Затем игрок берёт сколько угодно карт из колоды по одной, при этом он не может отказаться от карты, если уже взял её. Игрок не видел достоинства карт, взятых крупье, но видел их количество. Также он знает описанный выше алгоритм, согласно которому действует крупье.

Цель игрока — набрать карты с суммой их достоинств не больше S, но больше, чем у крупье, в этом случае он побеждает и получает один доллар. Если суммы достоинств карт у крупье и у игрока совпадают, то объявляется ничья, и игрок не получает ничего. Иначе игрок проигрывает и теряет один доллар.

В последнее время дела у казино идут не очень хорошо, поэтому Евгений решил тщательно изучить данную игру. Для начала он решил найти математическое ожидание выигрыша игрока при игре по оптимальной стратегии.

Формат входных данных

В первой строке даны два целых числа n и S ($2 \leqslant n \leqslant 50$, $2 \leqslant S \leqslant 400$) — количество различных типов карт и ограничение на сумму карт.

Во второй строке даны n целых различных чисел a_1, a_2, …, a_n ($2 \leqslant a_i \leqslant 400$) — достоинства типов карт.

В третьей строке даны n вещественных чисел p_1, p_2, …, p_n ($0.0001 \leqslant p_i \leqslant 1.0000$) — вероятности появления типов карт с достоинствами a_1, a_2, …, a_n соответственно. Гарантируется, что каждое из p_i дано не более чем с четырьмя знаками после десятичной точки, а сумма всех p_i равна единице.

Формат выходных данных

Выведите одно вещественное число — математическое ожидание выигрыша игрока при игре по оптимальной стратегии. Абсолютная или относительная погрешность должна не превосходить 10^{-6}.

Примеры

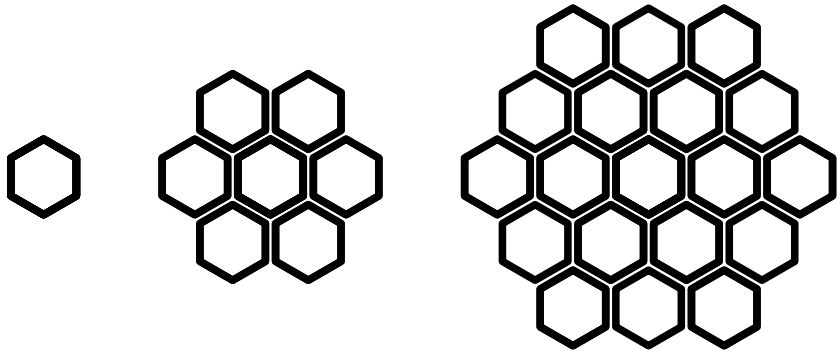
blackjack.in	blackjack.out
2 4 2 4 0.25 0.75	0.0351562500
2 6 2 4 0.25 0.75	-0.2614746094

Задача B. Колпачки и тортики

Имя входного файла:	caps-and-cakes.in
Имя выходного файла:	caps-and-cakes.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В детском саду «Учись, играя» дети каждый день ходят на прогулку в парк. Там они останавливаются перекусить: каждый получает одну порцию вкусного фруктового пюре в пакетике с колпачком. Пюре съедается, пакетик выбрасывается в урну, а шестиугольный колпачок остаётся у ребёнка. Колпачки эти не простые: они имеют форму одинаковых правильных шестиугольников с креплениями на сторонах, так что их можно составлять вместе как конструктор.

Сегодня шесть друзей из детского сада встретились на празднике и принесли каждый по n колпачков, чтобы поиграть в тортики. Тортик — это плоская фигура, составленная из колпачков и похожая на правильный шестиугольник. Тортик характеризуется своим размером — целым положительным числом. Тортик размера 1 — это просто одиноко лежащий колпачок. Тортик размера $s > 1$ собирается так: берётся тортик размера $(s - 1)$, после чего вдоль его границы добавляется новый слой колпачков. На рисунке представлены тортики размеров 1, 2 и 3.



Шестеро друзей хотят узнать, как, используя все колпачки, что у них есть, собрать наименьшее возможное количество тортиков. Помогите им это сделать.

Формат входных данных

В первой строке записано целое число n — количество колпачков, которые принёс с собой на праздник каждый из шести друзей ($1 \leqslant n \leqslant 10^{18}$).

Формат выходных данных

В первой строке выведите одно число k — количество тортиков. Это число должно быть как можно меньше.

Во второй строке выведите k чисел, разделяя их пробелами — размеры тортиков в любом порядке. Если решений с одинаковым k несколько, выведите любое из них.

Пример

caps-and-cakes.in	caps-and-cakes.out
3	6 1 2 1 2 1 1

Пояснение к примеру

В примере у всех друзей вместе оказалось 18 колпачков. Можно собрать из них шесть тортиков: четыре тортика размера 1 и два тортика размера 2. Используя все колпачки, меньше тортиков получить не удастся.

Задача C. Окружности

Имя входного файла:	`circles.in`
Имя выходного файла:	`circles.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дано n случайных окружностей. Постройте окружность, пересекающую максимально возможное число данных.

Формат входных данных

В первой строке записано целое число n — количество окружностей ($1 \leqslant n \leqslant 80$).

В следующих n строках записано по три целых числа x_i, y_i, r_i — координаты центра i-й окружности и её радиус.

Гарантируется, что во всех тестах, кроме примера, $n = 80$. Кроме того, все они сгенерированы следующим образом. Сначала выбираются случайные целые числа a_0, c, d ($1 \leqslant a_0, c, d \leqslant 10^9 + 7$). Далее генерируется последовательность a_i длины $3 \cdot n$ по правилу $a_i = (a_{i-1} \cdot c + d) \bmod (10^9 + 7) + 1$. После этого координаты центров и радиусы окружностей определяются как $x_i = a_{3 \cdot i - 2}$, $y_i = a_{3 \cdot i - 1}$, $r_i = a_{3 \cdot i}$.

Формат выходных данных

В первой строке выведите одно целое число k — число данных окружностей, пересекающихся с найденной вами окружностью.

Во второй строке выведите три вещественных числа — координаты центра и радиус построенной вами окружности.

Ваш ответ будет считаться корректным, если k максимально возможное и кольцо с центром (x, y) и радиусами $r - 10^{-9}, r + 10^{-9}$ внутренней и внешней окружностей соответственно пересекает ровно k данных окружностей

Мы гарантируем, что любое кольцо ширины 10^{-6} будет пересекать не больше окружностей, чем оптимальный ответ.

Пример

circles.in	circles.out
3 1 1 1 3 1 1 1 3 1	3 2 2 2.41421356237

Задача D. Си-плюс-минус

Имя входного файла:	`cpm.in`
Имя выходного файла:	`cpm.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мебибайт

Условие на четыре страницы?!

Начинающий участник

Это техника, надо сесть и написать

Опытный участник, не прочитавший условие

А как это написать?

Опытный участник, прочитавший условие

Современные среды разработки умеют многое. Например, мгновенно подсказывать подходящие по типу функции и переменные, показывать нужные отрывки из документации, искать ошибки, приводить код в соответствие со стилем, занимать всю доступную оперативную память... В этой задаче вам предстоит реализовать небольшой кусок функциональности среды разработки для языка $C\pm$.

Исходно у вас имеется программа S_0 на языке $C\pm$. Ваша программа должна по некоторым правилам, описанным ниже, отформатировать код этой программы при помощи добавления и удаления пробелов и/или переводов строк, получив тем самым программу F_0, которую надо вывести.

После этого вам требуется выполнить m запросов, которые могут менять программу. Обозначим программу, получившуюся после i-го запроса, как S_i ($1 \leqslant i \leqslant m$). Для каждой программы S_i путём форматирования генерируется программа F_i. После каждого запроса на модификацию вам требуется вывести одно число: $L(F_i)$, где $L(x)$ обозначает количество строк в программе x (ниже написано точное определение). А за $|x|$ мы далее будем обозначать длину x.

Запросы бывают следующих видов:

- Запрос «`add` p `@`t`@`» вставляет подстроку t в программу S_{i-1}, начиная с символа p:

$$S_i = S_{i-1,1}S_{i-1,2}\dots S_{i-1,p-1}t_1t_2\dots t_{|t|-1}t_{|t|}S_{i-1,p}S_{i-1,p+1}\dots S_{i-1,|S_{i-1}|}$$

- Запрос «`del` p l» удаляет из S_{i-1} подстроку длины l, начиная с символа p:

$$S_i = S_{i-1,1}S_{i-1,2}\dots S_{i-1,p-1}S_{i-1,p+l}S_{i-1,p+l+1}\dots S_{i-1,|S_{i-1}|}$$

- Запрос «`get` r s k» никак не меняет программу ($S_i = S_{i-1}$), однако вам требуется вывести некоторую подстроку программы F_i. Искомая подстрока характеризуется номером строки r, номером первого искомого символа в этой строке s и длиной искомой подстроки k.

  В случае, если в F_i отсутствует строка с номером r, ответом на запрос `get` считаем пустую строку. В случае, если в F_i строка с номером r присутствует, но в ней нет некоторых символов с номерами $s, s+1, \dots, s+k-1$, то эти символы считаются равными символу «`#`» (октоторп, ASCII-код 35).

Не гарантируется, что $S_0, S_1, \dots, S_n$ являются корректными программами на языке $C\pm$. Однако описанные ниже правила форматирования однозначно описывают требуемое поведение для произвольных программ.

Перед определением правил форматирования мы введём несколько дополнительных понятий:

- *Идентификатор* — это максимальная по включению последовательность латинских букв, цифр, квадратных скобок «`[`» и «`]`» (ASCII-коды 91 и 93), точек «`.`» (ASCII-код 46) и нижнего подчёркивания «`_`» (ASCII-код 95). Примеры корректных идентификаторов в языке $C\pm$: `hi`, `hello_world`, `bob[er9]5`, `12c.hairs`, `12_3`.

- *Бинарный оператор* — это один из следующих символов:

Символ	ASCII-код
`+`	43
`-`	45
`*`	42
`/`	47
`<`	60
`>`	62
`=`	61
`&`	38
`|`	124

- В языке $C\pm$ *разрешены символы* идентификаторов, бинарных операторов, круглые скобки («`(`», ASCII-код 40 и «`)`», ASCII-код 41), фигурные скобки («`{`», ASCII-код 123 и «`}`», ASCII-код 125), запятая «`,`» (ASCII-код 44), точка с запятой «`;`» (ASCII-код 59), пробел (ASCII-код 32) и символ новой строки (ASCII-код 10, в современных языках программирования обычно обозначается «\n»). Гарантируется, что во всех программах $S_0, \ldots, S_n$ будут встречаться только разрешённые символы.

- *Лексема* — это либо идентификатор, либо любой другой разрешённый символ, кроме пробела и перевода строки. Например, в программе `hello+ world` присутствуют три лексемы: `hello`, `+` и `world`.

- *R-балансом скобок некоторого вида* (круглых или фигурных) в некоторой строке S называется результат работы следующего псевдокода (приведён для круглых скобок, для фигурных — аналогично):

```
B <- 0
for each C in S do {
  if C is "(" then B <- B + 1
  if C is ")" then B <- B - 1
  if B < 0 then B <- 0
  if B > R then B <- R
}
return B
```

  Нестрого говоря, этот код вычисляет разность между количеством открывающих и закрывающих скобок определённого вида в S, но если этот баланс в некоторый момент «вылезает» за пределы отрезка $[0, R]$, то он «обрезается» (мы считаем, что R — либо положительное число, либо ∞). Например, в строке «`{{{){{}})}}(`» ∞-баланс и круглых, и фигурных скобок равен 1, а в строке «`(((((()`» 4-баланс круглых скобок равен 3.

- Позиция p находится *внутри круглых скобок*, если в подстроке t, заканчивающейся в p и начинающейся после последней фигурной скобки до p (или же в начале программы, если фигурных скобок до p нет), 4-*баланс круглых скобок* строго положителен. Например, если $t =$`(( ((( )))`, то p находится *внутри круглых скобок* (так как 4-баланс равен 1), а если $t =$`(( ((( ))) )`, то не находится (так как 4-баланс равен 0).

Форматирование производится следующим образом:

1. Вначале из программы удаляются все пробелы и переводы строк, при этом соседние лексемы в одну не склеиваются.

2. Далее добавляются переводы строк в следующих местах (если только из-за этого не образуются пустые строки):

   - После открывающей фигурной скобки «`{`».
   - Перед закрывающей фигурной скобкой «`}`».
   - После закрывающей фигурной скобки «`}`», только если сразу после неё не идёт идентификатор «`else`» или точка с запятой.
   - После точки с запятой «`;`», только если она не находится *внутри круглых скобок*.
   - В самом конце программы.

   Строкой в получившейся программе мы считаем последовательность символов, заканчивающуюся символом перевода строки. Таким образом, $L(x)$ определяется как количество символов перевода строки.

3. Затем в каждой строке добавляются пробелы между соседними лексемами в соответствии со следующей таблицей (плюс на пересечении строки A и столбца B обозначает добавление пробела между A и B):

	Идентификатор	Бинарный оператор	,	{	(	)	;
Идентификатор	+	+	-	+	-	-	-
Бинарный оператор	+	-	+	+	+	+	-
,	+	+	-	+	+	-	+
}	+	+	-	+	+	-	-
(	-	+	-	-	-	-	-
)	+	+	-	+	+	-	-
;	+	+	-	+	+	-	+

   Обратите внимание, что после открывающей фигурной скобки `{` никаких других символов в строке быть не может. Аналогично, не может быть символов перед закрывающей фигурной скобкой `}`.

4. Если после идентификатора `if`, `for`, `while`, `do` стояла круглая открывающая скобка, то между ними добавляется пробел.

5. После этого высчитывается *отступ* для каждой строки, равный ∞-*балансу фигурных скобок* во всех предыдущих строках.

6. В начале каждой строки дописывается количество пробелов, равное удвоенному *отступу* этой строки.

Формат входных данных

Первые строки входного файла содержат исходную программу S_0 в формате `@`S_0`@` (собака имеет ASCII-код 64). S_0 содержит только *разрешённые* символы ($1 \leqslant |S_0| \leqslant 10^5$). Обратите внимание, что S_0 не может содержать символ собаку, но может содержать пробелы и переводы строк. После второй собаки следует перевод строки.

Следующая строка содержит одно целое число m — количество запросов ($0 \leqslant m \leqslant 10^5$). Следующие строки содержат запросы, отделённые как минимум одним символом перевода строки: i-й запрос имеет следующий формат:

- `add` p `@`t`@` — добавить непустую подстроку t в текущую программу, начиная с позиции p ($1 \leqslant p \leqslant |S_{i-1}| + 1$). Гарантируется, что в подстроке t будут встречаться только *разрешённые* символы. В частности, в t могут встречаться пробелы и символы перевода строки. После второй собаки следует перевод строки. Гарантируется, что сумма $|t|$ по всем запросам `add` не превосходит 10^5.
- `del` p k — удалить из текущей программы подстроку длины k, начиная с символа p ($1 \leqslant p \leqslant p + k - 1 \leqslant |S_{i-1}|$).

- `get` r s k — запрос получения k символов строки r текущей программы, начиная с символа s ($1 \leqslant r, s, k \leqslant 10^5$). Гарантируется, что сумма k по всем запросам `get` не превосходит 10^5.

Пожалуйста, изучите пример для лучшего понимания формата ввода.

Формат выходных данных

В начале выходного файла выведите одно число: количество строк в отформатированной программе F_0 ($L(F_0)$). Затем выведите пробел, собаку, потом первые $\min(10^5, |F_0|)$ символов F_0, потом ещё одну собаку и символ перевода строки. Далее для i-го запроса выведите в отдельной строке:

- Для запросов `add` или `del`: количество строк в программе после соответствующего изменения ($L(F_i)$).
- Для запросов `get`: строку `@`t`@`, где t — искомая подстрока длины l отформатированной программы F_i. Если строка с номером r отсутствует в F_i (то есть $r > L(F_i)$), то выведите вместо t пустую строку. Если же $r \leqslant L(F_i)$, но какие-то из запрошенных символов не существуют в отформатированной программе, выведите вместо них символ «#» (ASCII-код 35). Считаем, что символ перевода строки в t не входит.

Пожалуйста, изучите пример для лучшего понимания формата вывода. Следуйте формату максимально строго: ваш ответ должен побайтово совпадать с требуемым.

Пример

cpm.in	cpm.out

`@if (a) { foo } else { bar; };`
`for (int a = 0; a< 5; a+=1)`
`{ some strange(command, wow); }}`
`else int foo bar {`
`  {(hi; strange; world)}`
 	
`   (hi; strange; world() {there;}`
`}@`
`14`
`get 3 2 10`
`get 100 1 1`
`add 32 @NEW`
`LINE@`
`get 6 1 30`
`del 32 8`
`add 18 @se if @`
`get 2 1 10`
`get 3 1 10`
`get 4 1 10`
`del 20 1`
`get 2 1 10`
`get 3 1 10`
`get 4 1 10`
`get 5 1 10`
	`16 @if (a) {`
	`  foo`
	`} else {`
	`  bar;`
	`};`
	`for (int a = 0; a < 5; a += 1) {`
	`  some strange(command, wow);`
	`}`
	`} else int foo bar {`
	`  {`
	`    (hi; strange; world)`
	`  }`
	`  (hi; strange; world() {`
	`    there;`
	`  }`
	`}`
	`@`
	`@ else {###@`
	`@@`
	`16`
	`@fNEW LINEor(int a = 0; a < 5; @`
	`16`
	`16`
	`@  foo#####@`
	`@} else if @`
	`@  bar;####@`
	`17`
	`@  foo#####@`
	`@}#########@`
	`@elseif se @`
	`@  bar;####@`

Задача E. Магараджа

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

Это интерактивная задача.

Магараджей называется фигура сказочных шахмат, которая может ходить либо как ферзь, либо как конь. Мобильность магараджи делает его очень сильной фигурой. Например, в игре с одноимённым названием «Магараджа» белый магараджа в одиночку сражается с полной армией чёрных фигур (которых назвают *сипаи*). Чтобы выиграть игру, магараджа должен поставить мат чёрному королю.

Вася услышал, что в игре «Магараджа» существует выигрышная стратегия за чёрных, которая работает независимо от ходов белых. Он думает, что это нечестно, и сейчас пытается убрать с доски часть чёрных фигур, анализируя получающиеся игры. В первую очередь он решил попробовать самый простой пример: магараджа должен заматовать одинокого чёрного короля.

Васин друг Петя сказал, что эта игра ещё более нечестна. Петя утверждает, что он может заматовать одинокого чёрного короля магараджей вообще без какой-либо информации о ходах и позиции короля.

Ваша задача — повторить достижение Пети. Ваша программа будет играть за белого магараджу со «злым» интерактором. Это значит, что интерактор сможет выбрать произвольную корректную начальную позицию чёрного короля (то есть отличную от позиции магараджи и такую, что король не находится под шахом) и его корректную последовательность ходов не только перед началом игры, но и в любой момент игры позднее. Если существует хотя бы одна такая последовательность, в которой магараджа съеден королём, или королю поставлен пат, ваше решение будет сочтено неправильным. Помимо прочего, у вас всего 50 ходов, чтобы поставить мат невидимому королю, иначе вы также проигрываете (вспомните, что в классических шахматах 50 ходов без взятий или движений пешки считается ничьей). Магараджа ходит первым. Успехов!

Протокол взаимодействия

Игра начинается с задания начальной позиции магараджи. Вы должны прочитать позицию из одной строки стандартного потока ввода. Затем вам необходимо делать ходы. Каждый ход делается с помощью записи в стандартный поток вывода одной строки, содержащей позицию, выбранную вами для магараджи. Интерактор отвечает одной строкой `-1`, если вы проиграли или сделали некорректный ход, `0`, если игра продолжается, и `1`, если вы выиграли. Ваша программа должна корректно завершить работу сразу же, как только она получила что-либо отличное от `0`.

Позиция — как во вводе, так и в выводе — всегда задаётся одной маленькой латинской буквой (`a`...`h`), за которой следует одна цифра (`1`...`8`).

Чтобы предотвратить буферизацию вывода, после каждого выведенного хода следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Пример

стандартный ввод	стандартный вывод
a1 0 -1	 b1 d2

Пояснение к примеру

Заметим, что в этом примере решение проиграло. Ваша задача — написать решение, которое выиграет.

Задача F. Матрёшки

Имя входного файла:	`matryoshka-dolls.in`
Имя выходного файла:	`matryoshka-dolls.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Матрёшка — это набор из деревянных кукол разных размеров. Все куклы, кроме самой маленькой, полые внутри и разбираются на две половинки: верхнюю и нижнюю. В собранном состоянии самая маленькая кукла лежит внутри второй по размеру, та — внутри третьей, и так далее.

У маленького Игната было несколько одинаковых матрёшек, каждая из которых состояла из n кукол. К сожалению, сейчас куклы рассыпаны по всему полу, а некоторые, возможно, вообще потерялись.

Теперь Игнат хочет собрать матрёшек, чтобы они не занимали на полу столько места. Поскольку некоторые куклы могли потеряться, при сборке он решил следовать более свободным правилам: матрёшка — это кукла, которая может либо быть пустой, либо содержать одну куклу любого меньшего размера — которая, в свою очередь, может либо быть пустой, либо содержать одну куклу ещё меньше, и так далее.

Известно, сколько кукол каждого размера находится на полу. Какое минимальное количество матрёшек может получиться у Игната по этим правилам?

Формат входных данных

В первой строке записано целое число n — количество размеров кукол ($1 \leqslant n \leqslant 100$). Во второй строке заданы через пробел n чисел a_1, a_2, …, a_n — количество кукол каждого размера ($1 \leqslant a_i \leqslant 100$). Размеры перечислены от больших к маленьким: например, a_1 — количество самых больших кукол, а a_n — самых маленьких.

Формат выходных данных

В первой строке выведите одно число k — минимально возможное количество матрёшек.

Пример

matryoshka-dolls.in	matryoshka-dolls.out	Иллюстрация
3 3 2 5	5	

Пояснение к примеру

Справа от примера нарисовано одно из возможных положений кукол, при котором матрёшек окажется 5.

Задача G. Не-крайнее значение

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

Это интерактивная задача.

Есть четыре попарно различных числа: `a`, `b`, `c` и `d`. Числа зафиксированы, но держатся в секрете. Можно задать не более трёх вопросов вида «верно ли, что одно из этих чисел меньше другого?» и после каждого вопроса сразу получить на него ответ. После этого следует назвать любое число, не являющееся ни наименьшим, ни наибольшим из этих четырёх чисел.

Напишите программу, которая будет решать эту задачу для любых `a`, `b`, `c` и `d`.

Протокол взаимодействия

Ваша программа должна общаться с программой жюри через стандартный поток ввода и стандартный поток вывода. Программа может задать от нуля до трёх вопросов, после чего сообщить ответ.

Принимаются вопросы в форме «верно ли, что $x < y$?». Чтобы задать такой вопрос, следует вывести строку «`Is` x `<` y`?`» в стандартный поток вывода. На месте переменных x и y должны стоять различные буквы из букв «`a`», «`b`», «`c`» и «`d`». Строка должна завершаться переводом строки.

Чтобы предотвратить буферизацию вывода, после каждого выведенного вопроса следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Ответ на вопрос в форме «`Yes.`» или «`No.`» в отдельной строке программа жюри сообщает в стандартный поток ввода.

Если ваша программа готова сообщить ответ, следует сделать это, выведя строку «`Value` z `is non-extremal.`» в стандартный поток вывода. На месте переменной z должна стоять буква «`a`», «`b`», «`c`» или «`d`». Эта строка также должна завершаться переводом строки. После этого ваша программа должна сразу корректно завершить работу.

Пример

Обратите внимание: **слева** указан **вывод** программы участника, а **справа** — то, что она после этого получает **на вход**.

действия участника	ответы жюри
`Is a < b?`	`Yes.`
`Is b < c?`	`Yes.`
`Is c < d?`	`No.`
`Value b is non-extremal.`	

Задача H. Параллелограммы

Имя входного файла:	`parallelograms.in`
Имя выходного файла:	`parallelograms.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Маленький Игорь очень любит собирать геометрические фигуры. У Игоря есть n отрезков. Игорь ещё маленький, поэтому он не знает, что отрезки можно поворачивать. Он умеет только параллельно переносить их.

Сегодня Игорь узнал новую фигуру — параллелограмм. Конечно, он хочет собрать из своих отрезков как можно больше этих замечательных фигур. Каждый отрезок он будет использовать для построения не более одного параллелограмма. Каждая сторона каждого параллелограмма будет состоять только из одного отрезка. Кроме того, Игоря интересуют только фигуры положительной площади. Некоторые отрезки могут остаться неиспользованными.

Помогите Игорю!

Формат входных данных

В первой строке записано целое число n — количество отрезков у Игоря ($1 \leqslant n \leqslant 10^5$). В следующих n строках записано по четыре целых числа $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2}$ — координаты концов i-го отрезка ($-10^9 \leqslant x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2} \leqslant 10^9$).

Формат выходных данных

Выведите одно число k — максимальное количество параллелограммов, которые Игорь может собрать из имеющихся у него отрезков.

Пример

parallelograms.in	parallelograms.out
5 0 0 1 2 2 2 5 3 5 7 4 5 0 1 3 2 3 5 3 5	1

Задача I. Плакаты

Имя входного файла:	`posters.in`
Имя выходного файла:	`posters.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Во Всемирном Конгрессе принимают участие n делегатов. В большом зале приготовлен круглый стол, за который как раз могут сесть n человек. Но есть одна трудность: на стене над каждым из n мест за столом висит плакат. У плакатов разное содержание, а у делегатов разные предпочтения, и во избежание неловкостей — а может быть, даже скандалов — некоторых делегатов не следует сажать под некоторые плакаты.

Главный Распорядитель Конгресса получил от своих подчинённых полный набор ограничений. В этих ограничениях делегаты условно пронумерованы целыми числами от 1 до n. Места за столом тоже пронумерованы целыми числами от 1 до n. Каждое ограничение — это пара (d, s), означающая, что делегат с номером d не должен сидеть на месте с номером s.

Помогите Главному Распорядителю, указав ему все возможные рассадки делегатов по местам, при которых все ограничения выполнены. По опыту прошлых конгрессов можно совершенно точно утверждать, что возможных рассадок получится не более 1000.

Формат входных данных

В первой строке записаны через пробел два целых числа n и m — количество делегатов и количество ограничений ($1 \leqslant n \leqslant 100$, $0 \leqslant m \leqslant 10\,000$). Каждая из следующих m строк содержит два числа d_i и s_i, означающих, что делегат d_i не должен сидеть на месте s_i ($1 \leqslant d_i, s_i \leqslant n$). Гарантируется, что все пары (d_i, s_i) различны.

Формат выходных данных

В первой строке выведите одно число k — количество возможных рассадок. В каждой из следующих k строк выведите одну из этих рассадок: n целых чисел a_1, a_2, …, a_n, разделённых пробелами. Число a_i обозначает номер места, на которое предлагается посадить делегата с номером i. Каждая возможная рассадка должна быть выведена ровно один раз. Рассадки можно выводить в любом порядке.

Гарантируется, что в правильном ответе $0 \leqslant k \leqslant 1000$.

Пример

posters.in	posters.out
4 6 1 2 1 3 2 3 2 4 4 4 2 1	3 4 2 1 3 1 2 4 3 4 2 3 1

Пояснение к примеру

В примере второй делегат может сидеть только на месте номер 2. Если выбрать, кто из оставшихся делегатов сидит на месте номер 1, оказывается, что оставшиеся два места заполняются делегатами однозначно.

Задача J. Четверичный квадрат

Имя входного файла:	`quaternary.in`
Имя выходного файла:	`quaternary.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Хорошо известно, что можно записать любое число в четверичной системе счисления, используя только цифры 0, 1, 2 и 3.

Однажды Вася записал целое число в четверичной системе счисления, а затем решил вычислить на компьютере квадратный корень из этого числа. Однако он случайно прочитал в своей программе это число, как если бы оно было в десятичной записи, но не заметил, что сделал что-то не так, потому что корень в десятичной записи тоже оказался целым числом, состоящим в своей десятичной записи только из цифр 0, 1, 2 и 3!

Позже, когда Вася осознал свою ошибку, он назвал такие числа *четверичными квадратами*. Формально, *четверичный квадрат* — это число, десятичная запись которого состоит только из цифр 0, 1, 2 и 3 и не содержит лидирующих нулей, являющееся точным квадратом и такое, что квадратный корень из него тоже является целым числом, десятичная запись которого состоит только из цифр 0, 1, 2 и 3.

Ваша задача очень проста. Отсортируем все четверичные квадраты длины n в порядке возрастания. Найдите k-й из них (нумерация начинается с единицы).

Формат входных данных

Входные данные состоят из одного или нескольких тестовых случаев.

Единственная строка каждого тестового случая содержит два целых числа n и k ($1 \leqslant n \leqslant 40$, $k \geqslant 1$). Гарантируется, что k-й четверичный квадрат существует. Сумма всех значений n во входных данных не превосходит 239.

Последняя строка содержит два нуля.

Формат выходных данных

Выведите одну строку для каждого тестового случая — k-й четверичный квадрат длины n.

Пример

quaternary.in	quaternary.out
3 1 3 2 0 0	100 121

Задача K. Передача по частям

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

Это интерактивная задача.

Разведчица Саша хочет передать в штаб важное сообщение, скрытое в строке из маленьких букв английского алфавита. Сегодня используется следующий алгоритм шифрования: сообщение — это самая часто встречающаяся подстрока переданной строки, а если таких подстрок несколько — лексикографически максимальная из них.

Поскольку передача — дело опасное, Саша собирается разбить строку на k частей и передавать их по порядку — передавать очередную часть, как только представится возможность. Между передачей двух соседних частей может случиться всякое, и нельзя исключать, что после любой части дальнейшая передача будет невозможна. Поэтому Саша хочет проверить, что в таком случае сообщение не получится каким-нибудь катастрофическим.

По заданным k частям в порядке их следования в строке выведите k сообщений: что получится при расшифровке, если передача прекратится после первой, второй, ..., k-й части.

Строка s лексикографически больше строки t, если либо t — собственный префикс s, либо есть позиция, в которой s и t не совпадают, и для самой левой из таких позиций i верно $s_i > t_i$.

Протокол взаимодействия

Ваша программа должна общаться с программой жюри через стандартный поток ввода и стандартный поток вывода.

Сначала на вход поступает число k на отдельной строке ($1 \leqslant k \leqslant 10\,000$). Далее на вход последовательно поступают k частей строки. Каждая часть задана на отдельной строке, состоит из маленьких букв английского алфавита и имеет длину от одной до миллиона букв включительно. Кроме того, гарантируется, что суммарная длина строки — от одной до миллиона букв включительно. Каждая часть, кроме первой, поступает на вход, как только получен ответ, связанный с предыдущей частью.

После получения каждой из k частей сразу выведите строку: каким будет расшифрованное сообщение, если передача в этот момент оборвётся. Строка должна завершаться переводом строки.

Чтобы предотвратить буферизацию вывода, после каждого выведенного сообщения следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

После вывода k ответов ваша программа должна сразу корректно завершить работу.

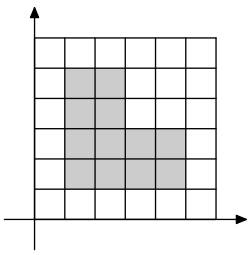
Пример

стандартный ввод	стандартный вывод
4	*(reading...)*
a	a
c	c
ba	a
cb	cb

Задача L. Водитель НЛО

Имя входного файла:	`ufo.in`
Имя выходного файла:	`ufo.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мебибайт

Водитель НЛО Арсений хочет посадить свой НЛО на площадку, имеющую форму квадрата $L \times L$ метров. НЛО Арсения имеет форму квадрата $2A \times 2A$ без одной из четвертей, расположенного относительно площадки так, как показано на рисунке. Здесь изображён случай $L = 6$, $A = 2$.



Для работы посадочного модуля на площадке введена система координат, оси которой направлены параллельно сторонам площадки, один из углов имеет координаты $(0, 0)$, а противоположный — (L, L).

Из-за технических особенностей посадочного модуля корабль можно посадить только так, чтобы все его углы находились в точках с целыми координатами, и, кроме того, корабль невозможно повернуть.

На площадке установлено несколько камер наблюдения. Каждая из них покрывает прямоугольник, содержащийся внутри площадки, и имеет некоторое качество картинки, выраженное целым числом q_i. *Заметностью* расположения НЛО Арсений называет самое большое из качеств картинок, на которых видно участок его корабля ненулевой площади. Заметность расположения, не попавшего ни под одну из камер, равна нулю. На рисунке ниже, соответствующем первому тесту из примера, для каждой клетки указана заметность расположения, если НЛО её заденет. Выделенный способ — единственный с заметностью 2.

Так как Арсений хочет, чтобы его НЛО оставался неопознанным, ему необходимо выбрать расположение посадки, заметность которого минимальна, при этом корабль должен целиком находиться внутри площадки. Помогите ему это сделать.

Формат входных данных

Входные данные состоят из одного или нескольких тестовых случаев, заданных друг за другом.

В первой строке описания теста даны два числа A и L ($1 \leqslant A$, $2 \cdot A \leqslant L \leqslant 10^9$), задающие размеры НЛО и площадки.

Во второй строке дано число n ($1 \leqslant n \leqslant 50\,000$) — количество камер наблюдения. В следующих n строках находится описание камер. Каждая камера задаётся пятью целыми числами $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2}$ и q_i ($0 \leqslant x_{i,1} < x_{i,2} \leqslant L$, $0 \leqslant y_{i,1} < y_{i,2} \leqslant L$, $1 \leqslant q_i \leqslant 10^9$) — координатами двух противоположных углов и качеством наблюдения.

Сумма n по всем тестовым случаям в одном тесте не превосходит $50\,000$.

Формат выходных данных

В ответ на каждый тестовый случай выведите две строки. В первой строке выведите одно число — минимально возможную заметность посадки. Во второй строке выведите два числа — координаты ближайшего к началу координат угла выбранного места посадки. Если возможных ответов несколько, выведите любой из них.

Пример

ufo.in	ufo.out	Пояснение
1 4 4 0 0 4 1 10 0 0 1 4 10 1 1 3 3 2 2 2 3 3 5 1 2 1 1 1 2 2 10	2 1 1 0 0 0