Без двух единиц подряд

Имя входного файла:	`fibseq.in`
Имя выходного файла:	`fibseq.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

По данному натуральному числу n выведите все двоичные последовательности длины n, не содержащие двух единиц подряд, в лексикографическом порядке.

Формат входных данных

Одно натуральное число n ($n \leqslant 20$).

Формат выходных данных

Каждая последовательность должна выводиться в отдельной строке, вывод должен завершаться символом новой строки. Числа, входящие в последовательность, должны быть разделены одним пробелом.

Примеры

fibseq.in	fibseq.out
3	0 0 0 0 0 1 0 1 0 1 0 0 1 0 1

Сочетания-1

Имя входного файла:	`comb1.in`
Имя выходного файла:	`comb1.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

По данным натуральным n и k выведите все двоичные последовательности длины n, содержащие ровно k единиц в лексикографическом порядке.

Формат входных данных

Входной файл содержит два числа, n и k ($1 \leqslant n \leqslant 100, 0 \leqslant k \leqslant n$).

Формат выходных данных

Каждая последовательность должна выводиться в отдельной строке, вывод должен завершаться символом новой строки. Числа, входящие в последовательность, должны быть разделены одним пробелом. Гарантируется, что количество чисел в выходном файле не превосходит 200000

Примеры

comb1.in	comb1.out
4 2	0 0 1 1 0 1 0 1 0 1 1 0 1 0 0 1 1 0 1 0 1 1 0 0

Сочетания-2

Имя входного файла:	`comb2.in`
Имя выходного файла:	`comb2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

По данным натуральным k и n ($1 \leqslant k \leqslant n$) выведите все **убывающие** последовательности длины k состоящие из чисел $1 \ldots n$ в лексикографическом порядке.

Формат входных данных

Во входном файле два числа — k и n ($1 \leqslant k \leqslant n \leqslant 1000$).

Формат выходных данных

Каждая последовательность должна выводиться в отдельной строке, вывод должен завершаться символом новой строки. Числа, входящие в последовательность, должны быть разделены одним пробелом. Гарантируется, что количество чисел в выходном файле не превосходит 500 000.

Примеры

comb2.in	comb2.out
3 5	3 2 1 4 2 1 4 3 1 4 3 2 5 2 1 5 3 1 5 3 2 5 4 1 5 4 2 5 4 3

Правильные скобочные последовательности

Имя входного файла:	`brackets.in`
Имя выходного файла:	`brackets.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дано натуральное число n. Выведите все правильные скобочные последовательности, состоящие из n открывающих круглых скобок и n закрывающих скобок в лексикографическом порядке.

Формат входных данных

Во входном файле одно число — n ($1 \leqslant n \leqslant 11$).

Формат выходных данных

Каждая последовательность должна выводиться в отдельной строке, вывод должен завершаться символом новой строки.

Примеры

brackets.in	brackets.out
3	((()))
	(()())
	(())()
	()(())
	()()()

Генерация правильных скобочных последовательностей - 2

Имя входного файла:	`brackets2.in`
Имя выходного файла:	`brackets2.out`
Ограничение по времени:	6 секунд
Ограничение по памяти:	64 мегабайта

По данному числу n выведите все правильные скобочные последовательности из круглых и квадратных скобок длины n. Придерживайтесь следующего порядка скобок: "([)]"(см. тест из условия)

Формат входных данных

Одно целое число n ($0 \leqslant n \leqslant 16$).

Формат выходных данных

Выведите все правильные скобочные последовательности из круглых и квадратных скобок длины n в лексикографическом порядке. Каждая послеждовательность должна выводиться в новой строке.

Примеры

brackets2.in	brackets2.out
4	(()) ([]) ()() ()[] [()] [[]] []() [][]

Разбиения на слагаемые

Имя входного файла:	`partition.in`
Имя выходного файла:	`partition.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Перечислите все разбиения целого положительного числа N на целые положительные слагаемые. Разбиения должны обладать следующими свойствами:

- Слагаемые в разбиениях идут в невозрастающем порядке.
- Разбиения перечисляются в лексикографическом порядке.

Формат входных данных

Во входном файле находится единственное число N ($1 \leqslant N \leqslant 40$).

Формат выходных данных

В выходной файл выведите искомые разбиения по одному на строку.

Примеры

partition.in	partition.out
4	1 1 1 1 2 1 1 2 2 3 1 4

Монетки

Имя входного файла:	`coins.in`
Имя выходного файла:	`coins.out`
Ограничение по времени:	2.2 секунды
Ограничение по памяти:	64 мегабайта

В Волшебной стране используются монетки достоинством $A_1, A_2, \ldots, A_M$. Волшебный человечек пришел в магазин и обнаружил, что у него есть ровно по две монетки каждого достоинства. Ему нужно заплатить сумму N. Напишите программу, определяющую, сможет ли он расплатиться без сдачи.

Формат входных данных

Сначала вводится целое число N ($1 \leqslant N \leqslant 10^9$), затем — целое число M ($1 \leqslant M \leqslant 10$) и далее M попарно различных целых чисел $A_1, A_2, \ldots, A_M$ ($1 \leqslant A_i \leqslant 10^9$).

Формат выходных данных

Выведите сначала K — количество монет, которое придется отдать Волшебному человечку, если он сможет заплатить указанную сумму без сдачи. Далее выведите K чисел, задающих достоинства монет. Если решений несколько, выведите вариант, в котором Волшебный человек отдаст наименьшее возможное количество монет. Если таких вариантов несколько, выведите любой из них.

Если без сдачи не обойтись, то выведите одно число 0. Если же у Волшебного человечка не хватит денег, чтобы заплатить указанную сумму, выведите одно число –1 (минус один).

Примеры

coins.in	coins.out
5 2 1 2	3 1 2 2
7 2 1 2	-1
5 2 3 4	0

Перестановка по номеру

Имя входного файла:	`bynumber.in`
Имя выходного файла:	`bynumber.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Демиурги Шамбамбукли и Мазукта уже достигли невероятного мастерства в искустве сотворения миров. Но им, как никому другому, известно, что нет предела совершенству. Неудивительно, что время от времени они собираются в на скорую руку сотворённом кафе за чашечкой кофе, чтобы вместе постигать вершины своего искуства. По своему немалому опыту они знают, что самые интересные и сложные закономерности проще всего обнаружить на самых простых примерах.

В этот раз они решили посмотреть, что выйдет из миров, в основу которых положены самые обычные перестановки. Чтобы не упустить ни один из возможных вариантов, демиурги решили использовать перестановки для сотворения миров по очереди, в лексикографическом порядке.

Шамбамбукли и Мазукта уже успешно создали $K - 1$ мир, но тут их настиг творческий кризис. Теперь только вы можете помочь им создать очередной мир и вдохновить их этим поступком на завершение эксперимента. По счастливому совпадению, этот мир как раз должен оказаться Хрымбелем, который демиурги в прошлый раз так и не смогли сотворить.

Формат входных данных

В первой строке входного файла записано число N ($1 \leqslant N \leqslant 12$) — количество элементов в перестановке, которая должна быть положена в основу Хрымбеля. Во второй строке число K ($1 \leqslant K \leqslant N!$) — номер перестановки.

Формат выходных данных

В выходной файл вывести N чисел через пробел — первооснову Хрымбеля.

Примеры

bynumber.in	bynumber.out
3 1	1 2 3

Номер по перестановке

Имя входного файла:	`perm.in`
Имя выходного файла:	`perm.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Дана перестановка из n чисел от 1 до n. Требуется найти её номер в лексикографическом порядке.

Формат входных данных

Во входном файле сначала записано число n ($1 \leqslant n \leqslant 12$). В следующей строке записана сама перестановка — n чисел, разделённых пробелами.

Формат выходных данных

В выходной файл нужно вывести единственное число — номер перестановки в лексикографическом порядке.

Примеры

perm.in	perm.out
3 2 1 3	3

Следующая перестановка

Имя входного файла:	`nextperm.in`
Имя выходного файла:	`nextperm.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Найдите следующую перестановку. Лексикографически первая перестановка является следующей для обратной.

Формат входных данных

В первой строке входного файла записано число N ($1 \leqslant N \leqslant 100\,000$) — количество элементов в перестановке. Во второй строке записана перестановка из N чисел.

Формат выходных данных

В выходной файл вывести N чисел — искомую перестановку.

Примеры

nextperm.in	nextperm.out
3 3 2 1	1 2 3
2 1 2	2 1

Предыдущая перестановка

Имя входного файла:	`prev.in`
Имя выходного файла:	`prev.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Найдите предыдущую в лексикографическом порядке перестановку. Перестановка вида $N, N-1, ..., 3, 2, 1$ является предыдущей для $1, 2, 3, ..., N-1, N$

Формат входных данных

В первой строке входного файла записано число N ($1 \leqslant N \leqslant 10^5$) количество элементов в перестановке. Во второй строке записана перестановка.

Формат выходных данных

В выходной файл вывести N чисел — искомую перестановку.

Примеры

prev.in	prev.out
3 1 2 3	3 2 1

Следующее сочетание

Имя входного файла:	`nextcomb.in`
Имя выходного файла:	`nextcomb.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Дано множество целых чисел от 1 до n. Рассмотрим подмножество этого множества, состоящее из k элементов, в возрастающем порядке.

Выведите следующее в лексикографическом порядке подмножество из k элементов.

Формат входных данных

В первой строке входного файла содержатся целые положительные числа n и k ($1 \leqslant k \leqslant n \leqslant 50$). Во второй строке содержится k целых чисел от 1 до n в возрастающем порядке — подмножество из k элементов.

Формат выходных данных

Выведите следующее в лексикографическом порядке после данного подмножество из k элементов. Если следующего подмножества нет, выведите 0.

Примеры

nextcomb.in	nextcomb.out
6 4 1 4 5 6	2 3 4 5
6 2 5 6	0

30 кресел

Имя входного файла:	`choose.in`
Имя выходного файла:	`choose.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Потерпев фиаско в погоне за стульями Остап не пал духом, а ввязался в новую авантюру. Услышав, что неподалёку на аукционе распродаются n старинных кресел, не долго думая он решил попытать удачу и проверить: не скрываются ли сокровища в одном из них. Придя на торги, Остап понял, что денег у него хватит на выкуп ровно k из n кресел. Своим самым счастливым числом Остап считает число m, поэтому он снова обращается к Вам за помощью и просит выбрать m-е сочетание k из n кресел.

Формат входных данных

Во входном файле заданы числа n, k и m. $1 \leqslant k \leqslant n \leqslant 30$, $0 \leqslant m \leqslant \binom{n}{k} - 1$.

Формат выходных данных

Выведите в выходной файл в возрастающем порядке номера кресел, входящие в m-е в лексикографическом порядке сочетание по k из чисел от 1 до n. Сочетания занумерованы, начиная с 0.

Примеры

choose.in	choose.out
4 2 3	2 3

Следующее разбиение на слагаемые

Имя входного файла:	`nextpartition.in`
Имя выходного файла:	`nextpartition.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Формат входных данных

Входной файл содержит одну строку — разбиение числа n на слагаемые ($1 \leqslant n \leqslant 100\,000$). Слагаемые в разбиении следуют в неубывающем порядке.

Формат выходных данных

Выведите в выходной файл одну строку — разбиение числа n на слагаемые, следующее в лексикографическом порядке после приведенного во входном файле. Если во входном файле приведено последнее разбиение n на слагаемые, выведите `No solution`.

Примеры

nextpartition.in	nextpartition.out
5=1+1+3	5=1+2+2
5=5	No solution

Как убить время

Имя входного файла:	`num2part.in`
Имя выходного файла:	`num2part.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Некто Симон уже давно дорешал все задачи из контестов и теперь убивает время на практике тем, что раскладывает найденные у комповника n камешков в кучки разного размера. Изначально он разложил все камешки по одному. В каждую следующую минуту он выкладывает из них следующее лексикографическое разбиение на кучки. Порядок разбиений будем определять следующим образом: рассмотрим все разбиения n камешком на кучки, в каждом разбиении упорядочим их в порядке не убывания. Будем считать, что разбиение $a_1+a_2+...+a_n$ лексикографически меньше $b_1+b_2+...+b_m$, если для некоторого k $\forall j \leqslant k : a_j = b_j$ и либо $k = n$, либо $a_{k+1} < b_{k+1}$.

Формат входных данных

Во входном файле заданы числа n и r. $1 \leqslant n \leqslant 100$, разбиение с номером r — существует.

Формат выходных данных

Выведите r-ое разбиение n камешков на кучки, разбиения нумеруются с 0.

Примеры

num2part.in	num2part.out
4 3	2+2
5 5	2+3