Порождение комбинаторных объектов. Итеративные алгоритмы.

Во всех задачах этого листка решения должны быть нерекурсивными. Время работы (если не указано иное) должно быть пропорционально количеству выводимых на экран чисел.

В задачах $A-S$ не допускается использование вспомогательных массивов (единственный массив используется для хранения результата).

Определения

Лексикографический порядок: $(x_1, \ldots, x_n) < (y_1, \ldots, y_n) \Leftrightarrow \exists k \geqslant 1 : x_k < y_k, x_i = y_i, \forall i < k$

При сравнении последовательностей разной длины действует это же определение, но сначала короткая *мысленно* дополняется нужным количеством элементов, мѐньших любого элемента последовательности.

Обратный лексикографический порядок: меньше та последовательность, которая больше в лексикографическом порядке.

A. *Все последовательности длины N из чисел 1...K*

Напечатать все последовательности длины N из чисел $1 \ldots K$.

В первой строке вводится натуральное число N, во второй — натуральное число K.

Программа должна вывести все последовательности в лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
2 3	1 1 1 2 1 3 2 1 2 2 2 3 3 1 3 2 3 3

B. *Элемент не больше своего номера*

Напечатать все последовательности натуральных чисел длины N, у которых i-ый член не превосходит i.

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все последовательности в лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
3	1 1 1 1 1 2 1 1 3 1 2 1 1 2 2 1 2 3

C. *Все возрастающие*

Перечислить все строго возрастающие последовательности длины N из чисел $1 \ldots K$.

В первой строке вводится натуральное число N, во второй — натуральное число K ($K \geqslant N$).

Программа должна вывести все последовательности в лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
2 3	1 2 1 3 2 3

D. *Все убывающие*

Перечислить все убывающие последовательности длины N из чисел $1 \ldots K$.

В первой строке вводится натуральное число N, во второй — натуральное число K ($K \geqslant N$).

Программа должна вывести все последовательности в **обратном** лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
2 3	3 2 3 1 2 1

E. *Все подмножества*

Напечатать все подмножества множества $\{1 \dots N\}$.

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести $2^N - 1$ строк, в каждой из которых перечислены элементы одного из подмножеств (пустое подмножество печатать не надо). Подмножества разной мощности должны быть упорядочены по размеру в порядке возрастания количества элементов в подмножестве, подмножества равной мощности должны быть перечислены в лексикографическом порядке.

Элементы подмножеств должны быть разделены одним пробелом.

Input	Output
3	1 2 3 1 2 1 3 2 3 1 2 3

F° *Все перестановки*

Напечатать все перестановки чисел $1 \dots N$ (то есть последовательности длины n, в которые каждое из этих чисел входит по одному разу).

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все перестановки чисел $1 \dots N$ в лексикографическом порядке. Элементы перестановок должны быть разделены одним пробелом.

Input	Output
3	1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1

G. *Следующая анаграмма*

Для данного слова (последовательности строчных латинских букв) выведите следующее за ним (в лексикографическом порядке) слово, которое может быть получено из данного перестановкой букв (анаграмму). Если данное слово уже является последним среди всех своих анаграмм, то необходимо вывести первую возможную (в лексикографическом порядке) анаграмму.

Указание 1: для хранения последовательности символов используйте массив символов (массив можно изменять в отличие от строки).

Указание 2: поскольку количество слов во входных данных заранее неизвестно, то читать их требуется из файла.

Задана последовательность слов, по одному слову в строке. Длина одного слова не превышает 50 символов.

Необходимо вывести результат для каждого полученного на вход слова.

Input	Output
aab aba baa aaa	aba baa aab aaa

H* *Все вложения*

Перечислить все вложения (функции, переводящие разные элементы в разные) множества $\{1 \dots K\}$ в $\{1 \dots N\}$. Порождение очередного элемента должно требовать не более $C * K$ действий.

В первой строке вводится натуральное число K, во второй — натуральное число N ($N \geqslant K$).

Программа должна вывести все возможные последовательности длины K, состоящие из элементов множества N. Порядок следования последовательностей отличается от лексикографического (см. пример). Элементы указанных последовательностей должны быть разделены одним пробелом.

Input	Output
2 3	1 2 2 1 1 3 3 1 2 3 3 2

I. *Двоичные последовательности, без двух единиц подряд*

По данному натуральному N выведите все двоичные последовательности длины N, не содержащие двух единиц подряд.

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все указанные последовательности в лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
3	0 0 0 0 0 1 0 1 0 1 0 0 1 0 1

J* *Двоичные последовательности, не более K единиц*

По данным натуральным N и K ($0 \leqslant K \leqslant N, N \geqslant 1$) выведите все двоичные последовательности длины N, содержащие не более K единиц.

В первой строке вводится натуральное число N, во второй — натуральное число K ($K \leqslant N$).

Программа должна вывести все указанные последовательности в лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
3 1	0 0 0 0 0 1 0 1 0 1 0 0

K. *Двоичные последовательности, ровно K единиц*

По данным натуральным N и K ($0 \leqslant K \leqslant N, N \geqslant 1$) выведите все двоичные последовательности длины N, содержащие ровно K единиц.

В первой строке вводится натуральное число N, во второй — натуральное число K ($K \leqslant N$).

Программа должна вывести все указанные последовательности в лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
3 2	0 1 1 1 0 1 1 1 0

L* *Двоичные последовательности, не более K единиц без двух единиц подряд*

По данным натуральным N и K ($0 \leqslant K \leqslant N, N \geqslant 1$) выведите все двоичные последовательности длины N, содержащие не более K единиц без двух единиц подряд.

В первой строке вводится натуральное число N, во второй — натуральное число K.

Программа должна вывести все указанные последовательности в лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
3 2	0 0 0 0 0 1 0 1 0 1 0 0 1 0 1

M. *Двоичные последовательности, ровно K единиц без двух единиц подряд*

По данным натуральным $N \geqslant 0$ и $K \geqslant 0$ выведите все двоичные последовательности длины N, содержащие ровно K единиц без двух единиц подряд. Гарантируется, что существует хотя бы одна такая последовательность для данных N и K.

В первой строке вводится натуральное число N, во второй — натуральное число K.

Программа должна вывести все указанные последовательности в лексикографическом порядке. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
4 2	0 1 0 1 1 0 0 1 1 0 1 0

N. *Разбиения натурального числа на слагаемые: невозрастание, лексикографический порядок*

Перечислить в лексикографическом порядке все невозрастающие разбиения целого положительного числа N на целые положительные слагаемые (разбиения, отличающиеся лишь порядком слагаемых, считаются одинаковыми).

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все указанные последовательности. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
4	1 1 1 1 2 1 1 2 2 3 1 4

O. *Разбиения натурального числа на слагаемые: невозрастание, обратный лексикографический порядок*

Перечислить в обратном лексикографическом порядке все невозрастающие разбиения целого положительного числа N на целые положительные слагаемые (разбиения, отличающиеся лишь порядком слагаемых, считаются одинаковыми).

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все указанные последовательности. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
5	5 4 1 3 2 3 1 1 2 2 1 2 1 1 1 1 1 1 1 1

P. *Разбиения натурального числа на слагаемые: неубывание, лексикографический порядок*

Перечислить в лексикографическом порядке все неубывающие разбиения целого положительного числа N на целые положительные слагаемые (разбиения, отличающиеся лишь порядком слагаемых, считаются одинаковыми).

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все указанные последовательности. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
4	1 1 1 1 1 1 2 1 3 2 2 4

Q. *Разбиения натурального числа на слагаемые: неубывание, обратный лексикографический порядок*

Перечислить в обратном лексикографическом порядке все неубывающие разбиения целого положительного числа N на целые положительные слагаемые (разбиения, отличающиеся лишь порядком слагаемых, считаются одинаковыми).

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все указанные последовательности. Элементы последовательностей должны быть разделены одним пробелом.

Input	Output
5	5 2 3 1 4 1 2 2 1 1 3 1 1 1 2 1 1 1 1 1

R. *Правильные скобочные последовательности длины 2N*

Дано натуральное число N. Выведите все правильные скобочные последовательности, состоящие из N открывающихся круглых скобок и N закрывающихся скобок в лексикографическом порядке (открывающая скобка меньше закрывающей).

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все указанные скобочные последовательности в лексикографическом порядке (открывающая скобка меньше, чем закрывающая), не разделяя скобки пробелами.

Input	Output
3	((())) (()()) (())() ()(()) ()()()

S* *Правильные скобочные последовательности длины 2N вложенности не более K*

Даны натуральные числа N и K. Выведите в лексикографическом порядке все правильные скобочные последовательности, состоящие из N открывающихся круглых скобок и N закрывающихся скобок так, что максимальная вложенность не превосходит K.

В первой строке вводится натуральное число N, во второй — натуральное число K ($K \leqslant N$).

Программа должна вывести все указанные скобочные последовательности в лексикографическом порядке (открывающая скобка меньше, чем закрывающая), не разделяя скобки пробелами.

Input	Output
4 2	(()()()) (()())() (())(()) (())()() ()(()()) ()(())() ()()(()) ()()()()

T° *Код Грея*

Перечислить все последовательности длины N из чисел $1 \dots K$ в таком порядке, чтобы каждая следующая отличалась от предыдущей в единственной позиции, причем не более, чем на 1. Разрешается использовать вспомогательный массив длины N.

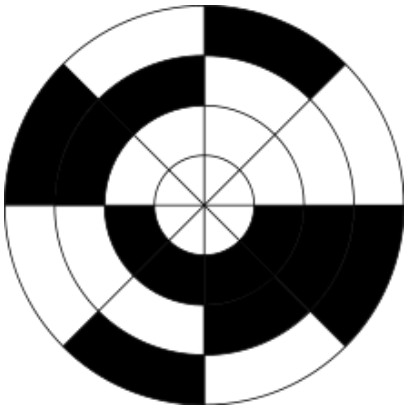
В первой строке вводится натуральное число N, во второй — натуральное число K.

Программа должна вывести все указанные последовательности по одной на строке. Элементы последовательностей должны быть разделены пробелами.

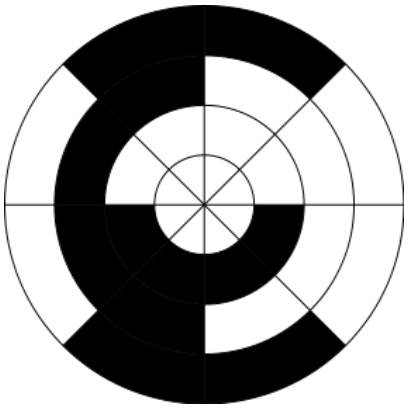
Input	Output
3 2	1 2 1 2 2 1 2 2 2 2 1 2 2 1 1 1 1 1 1 1 2 1 2 2

Терминологическое замечание: вообще говоря, кодом Грея называется именно бинарный код, то есть такой способ перечисления бинарных последовательностей, в котором соседние последовательности отличаются только в одной позиции. Имя Фрэнка Грея этот код получил благодаря его патенту 1953 года на алгоритм обработки телевизионного сигнала, где он использовал эту идею. Хотя сама идея была известна за 75 лет до этого, например, Эмилю Бодо (Jean-Maurice-Émile Baudot).

На картинке ниже приведён пример того, как код Грея может быть использован для устойчивого способа преобразования аналогового сигнала в цифровой. Например, нам требуется закодировать угол поворота диска. Разобьём диск на 2^N секторов и N дорожек (на картинках $N = 3$). В каждом секторе "кусочек" дорожки будет считывать сигнал, из совокупности этих сигналов складывается двоичное число, которое будет соответствовать углу поворота.



Проблема, правда, в том, что если вдруг случайно при переходе, например, от `011` к `100` внутренний датчик запоздает (1 соответствует закрашенной области, а 0 — незакрашенной), то прочитается сигнал `000`, что будет означать ошибку почти на развёрнутый угол, что для датчика измерения угла совершенно неприемлемая ошибка.



Иной способ кодирования, а именно такой, в котором соседние секторы отличаются лишь одним битом, более устойчив. Ошибка, если и случится, будет означать, что мы ошиблись на $\frac{360}{2^N}$ градусов.

U* *Перестановки: соседние отличаются одной транспозицией*

Напечатать все перестановки чисел $1 \dots N$ так, чтобы каждая следующая получалась из предыдущей перестановкой (транспозицией) двух соседних чисел.

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все перестановки в порядке, удовлетворяющем условию задачи.

Input	Output
3	2 1 3 1 2 3 1 3 2 3 1 2 3 2 1 2 3 1

V* *Все расстановки скобок в неассоциативном произведении*

Перечислить все расстановки скобок в произведении N сомножителей. Порядок сомножителей не меняется, скобки должны полностью определять порядок действий.

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести все способы расстановки скобок в приведённом ниже формате. Порядок вывода последовательностей может быть любым.

Input	Output
4	x*(x*(x*x)) x*((x*x)*x) (x*(x*x))*x (x*x)*(x*x) ((x*x)*x)*x

W* *N непересекающихся хорд*

На окружности задано $2N$ точек, пронумерованных от 1 до $2N$. Перечислить все способы провести N непересекающихся хорд с вершинами в этих точках.

На вход программе подаётся единственная строка, содержащая натуральное число N.

Программа должна вывести в произвольном порядке все наборы хорд, удовлетворяющие условию. Набор хорд описывается N парами чисел, записанных в одной строке. Каждая хорда в наборе должна быть записана в виде пары вершин, разделённых пробелом.

Input	Output
3	1 2 3 4 5 6 1 2 4 5 3 6 2 3 1 4 5 6 2 3 4 5 1 6 3 4 2 5 1 6

Комментарий к примеру

В первой строчке ответа имеется в виду следующий набор хорд:

$1-2, 3-4, 5-6$

X* *Триангуляция*

Перечислите все способы триангуляции выпуклого N-угольника, то есть все способы разрезать его на $N-2$ треугольника.

Программа должна по заданному числу N вывести группы пар чисел. Каждая группа соответствует одной триангуляции. Пары чисел в группе — номера вершин, которые надо соединить в результате триангуляции.

Input	Output
5	...

В решении следующих задач вам предлагается использовать на практике алгоритмы перебора, изученные выше. Вооообще говоря, некоторые задачи можно решить и иначе (притом более эффективно). Но об этом чуть позже...

Y. *Потратить все деньги*

Однажды Вовочке на день рожденья подарили S рублей. Он без ума от булочек, поэтому немедленно отправился в кондитерскую. Там продаются K разных булочек, для каждой известна её цена. Вовочка хочет потратить все деньги и не так уж важно, на какие именно булочки. Сможет ли он это сделать?

В первой строчке даны натуральные числа S, количество денег у Вовочки, и K — количество булочек в кондитерской. В следующей строке перечислены стоимости каждой булочки.

Если Вовочка не сможет купить ни одной булочки, выведите число 0. В противном случае надо вывести в порядке убывания два различных максимальных числа — суммы, которые он может потратить, и номера булочек, которые надо для этого купить.

Если сумма, которую удаётся потратить на булочки только одна, надо вывести её и номера соответствующих булочек.

Input	Output
100 10 5 15 25 35 45 55 65 75 85 95	100 0 9 95 9
10 4 19 9 13 15	9 1
10 4 11 11 11 11	0
10 4 3 5 13 15	8 0 1 5 1

Пояснение к первому тесту: Вовочка может потратить все деньги, купив булочки за 95 и 5 рублей. А также может потратить 95 рублей. Это две максимальные суммы, которые может потратить Вовочка. Обратите внимание — вам не надо выводить второе число, равное 100 (как сумму 45 и 55), так как требуется вывести две *различные* суммы.

Пояснение ко второму тесту: имеющихся у него денег Вовочке хватит лишь на одну булочку за 9 рублей.

Z. *Вовочка берётся за ум*

После того, как Вовочка потратил все деньги на булочки, ему стало грустно. Он решил, что станет лучше учиться и у него больше никогда не будет проблем с булочками. Для реализации этого плана он зарегистрировался на курсере (www.coursera.org), чтобы проходить он-лайн курсы. Однако курсы достаточно сложные, и Вовочка пока не способен слушать лекции два дня подряд. Для каждого из K дней Вовочка для себя определил ценность лекции в этот день. Помогите ему выбрать лекции с максимальной ценностью.

В первой строчке дано натуральное число K — количество дней. Во второй строчке даны K целых чисел — ценности лекций для Вовочки (некоторые лекции он считает вредными, потому их ценность отрицательная).

Требуется вывести пять лучших различных значений — суммарных ценностей лекций вместе с соответствующими им графиками посещения лекций. При этом запрещается слушать лекции два дня подряд. Выведите в порядке возрастания номера дней (индексы массива), в которые лучше всего слушать лекции. Если вариантов посещения лекций для достижения определённой суммарной стоимости несколько — выведите любой.

Input	Output
4 100 150 70 15	170 0 2 165 1 3 150 1 115 0 3 100 0
5 -10 100 130 20 -10	130 2 120 0 2 110 0 2 4 100 1 90 1 4

Пояснение к первому тесту: наибольшая ценность, которую Вовочка может получить от лекций равна 170, для этого надо посетить лекции в 0 и 2 дни. Затем следует ценность 165, для чего следует посетить лекции в дни с номерами 1 и 3. Далее приведены ещё три значения суммарной ценности.

Пояснение ко второму тесту: к сожалению, для получения пяти различных значений требуется посещать вредные лекции, потому Вовочке хочется хотя бы минимизировать вред.

ZA. *Вовочка и поднабор с нулевой суммой*

На курсе по алгоритмам Вовочке предлагалось решить следующую NP-полную задачу: Даны N чисел. Можно ли среди них выбрать K так, чтобы их сумма равнялась нулю?

Решите её и вы.

В первой строчке даны натуральные числа N и K. Во второй строчке даны N целых чисел. Выведите через пробел в порядке возрастания K номеров чисел, имеющих сумму 0. Если такого набора нет, то выведите слово `Impossible`.

Input	Output
3 2 100 -30 30	1 2
3 2 100 -30 -70	Impossible

ZB. *Вовочка и задача коммивояжёра*

Следующим шагом для продолжения обучения Вовочка решил выбрать для себя лучший университет. Он собрал список из $N+1$ лучшего университета (университет в городе Вовочки имеет номер 0). На страничке `awesome-python` он нашёл ссылку на пакет `robobrowser`, при помощи которого с известного сайта добыл стоимости перелётов из города с i-м университетом в город с j-м университетом для всех i и j.

Вовочка хочет найти самый дешёвый круговой маршрут для посещения всех ВУЗов.

В первой строчки дано число N — число университетов, не считая университета в родном городе Вовочке. В следующих $N+1$ строках дана таблица с ценами перелётов из города с i-м университетом в город с j-м. Выведите в первой строчке минимальную стоимость перелётов, а во второй — любую из последовательностей перелётов с минимальной стоимостью.

Input	Output
3 0 8000 6500 8000 8000 0 2000 1500 8000 1500 0 2000 6500 2000 2000 0	16000 0 2 1 3 0

ZC. *Всё по i*

Во время тура по университетам Вовочка обнаружил замечательную кондитерскую: «Всё по i». В ней для всех i от 1 до 30 были булочки за i долларов. Былая любовь к булочкам проснулась в нём, и он решил купить булочек на все оставшиеся S долларов.

В первой строчке дано натуральное число S — доступная Вовочке сумма денег. В следующей строчке даны $min(S, 30)$ натуральных чисел — привлекательность булочек каждой стоимости для Вовочки.

Выведите любой из лучших наборов булочек, которые можно купить за S долларов.

Input	Output
10 1 1 2 3 3 4 9 7 8 9	7 1 1 1