Задача A. Сумма двух

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

Формат входного файла

В первой строке входного файла расположены два целых числа A и B, не превосходящих 1 000 по модулю.

Формат выходного файла

Ваша программа должна выдавать в выходной файл одно число — сумму чисел A и B.

Примеры

sum.in	sum.out
2 3	5
17 -18	-1

Задача B. Сколько лестниц?

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

Назовём *лестницей длины L и высоты H* последовательность чисел a_0, a_1, a_2, …, a_L такую, что

- $a_0 = 0$, $a_L = H$;
- Для любого $0 < i \leqslant L$ либо $a_i = a_{i-1} + 1$, либо $a_i = a_{i-1} + 2$.

По заданным L и H найдите количество различных лестниц длины L и высоты H. Лестницы считаются различными, если соответствующие им последовательности различаются хотя бы в одном элементе.

Формат входного файла

В первой строке входного файла заданы два числа L и H через пробел ($1 \leqslant L, H \leqslant 15$).

Формат выходного файла

Выведите в выходной файл одно число — количество лестниц длины L и высоты H.

Примеры

stairs.in	stairs.out
1 1	1
2 3	2

Задача C. Поколение комбинаторов

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

Сочетанием из n элементов по k называется убывающая последовательность из k чисел из диапазона от 1 до n.

Сгенерируйте все сочетания из n элементов по k в антилексикографическом порядке, т.е. так, что для любых двух выведенных сочетаний первые l чисел равны, а $l + 1$-е в предыдущем больше, чем в следующем.

Формат входного файла

Во входном файле содержатся два целых числа n и k. $1 \leq k \leq n \leq 15$.

Формат выходного файла

В выходной файл выведите все сочетания из n элементов по k в антилексикографическом порядке, по одному сочетанию на строку.

Пример

generation.in	generation.out
3 2	3 2 3 1 2 1

Задача D. Сосчитайте...

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

Ваша задача — подсчитать количество неотрицательных целых решений неравенства

$$x_1 + x_2 + \ldots + x_m \leqslant n,$$

где $1 \leqslant m \leqslant 30$, $0 \leqslant n \leqslant 30$.

Формат входного файла

Входной файл состоит из двух целых чисел m и n.

Формат выходного файла

В выходной файл необходимо вывести количество решений этого неравенства в неотрицательных целых числах.

Пример

count.in	count.out
3 5	56

Задача E. Кубики

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

Маленький Леша очень любит играть с кубиками. Он бросал их и бросал, пока не придумал себе более интересное занятие. Леша написал на каждой грани своих кубиков цифры от 1 до 6 (на любом кубике каждая из этих цифр встречается ровно один раз). Теперь он бросает все свои m кубиков (после чего каждый из них падает какой-то гранью вверх) и хочет, чтобы число, записанное на верхних гранях кубиков, делилось на задуманное им натуральное число n. Получившееся на верхних гранях число читается слева направо таким образом, что старший разряд — это цифра, написанная на верхней грани первого кубика, а младший — цифра, написанная на верхней грани m-го. Посчитайте вероятность, с которой Леша добьется успеха при однократном бросании своих кубиков.

Формат входного файла

В первой строке входного файла заданы два числа: m ($1 \leqslant m \leqslant 7$) и n ($1 \leqslant n \leqslant 100000$).

Формат выходного файла

В выходной файл должно быть выдано одно число, записанное ровно с тремя знаками после запятой — вероятность успеха Лешиных действий.

Примеры

dice.in	dice.out
1 2	0.500
1 3	0.333

Задача F. Размещение с повторениями по номеру

Имя входного файла:	`arrange2.in`
Имя выходного файла:	`arrange2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Размещением с повторениями называется способ выбрать из n элементов k, при этом способы, отличающиеся перестановкой

элементов, считаются различными, а один и тот же элемент можно выбирать более одного раза. Например, существует всего 9 размещений с повторениями из 3 элементов по 2.

В этой задаче мы будем рассматривать все k-элементные размещения с повторениями множества из n чисел от 1 до n. Естественно, что все эти размещения можно упорядочить лексикографически как вектора. Скажем, при $n = 3$ и $k = 2$ список упорядоченных размещений с повторениями будет выглядеть так: $(1,1)$, $(1,2)$, $(1,3)$, $(2,1)$, $(2,2)$, $(2,3)$, $(3,1)$, $(3,2)$, $(3,3)$.

В этой задаче требуется найти размещение с повторениями по лексикографическому номеру (нумерация ведётся с нуля).

Формат входного файла

Три числа n, k и x ($1 \leqslant k \leqslant n \leqslant 15$), x задаёт номер существующего размещения с повторениями.

Формат выходного файла

Выведите k чисел, задающих требуемое размещение с повторениями.

Пример

arrange2.in	arrange2.out
3 2 1	1 2
4 2 5	2 2

Задача G. Номер по размещению

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

Размещением называется способ выбрать из n элементов k, при этом способы, отличающиеся перестановкой элементов, считаются различными. Например, существует всего 6 размещений из 3 элементов по 2.

В этой задаче мы будем рассматривать все k-элементные размещения множества из n чисел от 1 до n. Естественно, что все эти размещения можно упорядочить лексикографически как вектора. Скажем, при $n = 3$ и $k = 2$ список упорядоченных размещений будет выглядеть так: $(1,2)$, $(1,3)$, $(2,1)$, $(2,3)$, $(3,1)$, $(3,2)$.

В этой задаче требуется найти лексикографический номер по размещению (нумерация ведётся с нуля).

Формат входного файла

В первой строке даны два числа n, k ($1 \leqslant k \leqslant n \leqslant 20$). Во второй строке даны k чисел, задающих требуемое размещение.

Формат выходного файла

Выведите номер размещения.

Пример

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

Задача H. Сочетание с повторениями по номеру

Имя входного файла:	comb2.in
Имя выходного файла:	comb2.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Сочетанием с повторениями называется способ выбрать из n элементов k, при этом способы, отличающиеся перестановкой элементов, считаются одинаковыми, но один элемент можно выбирать более одного раза. Например, существует всего 6 сочетаний с повторениями из 3 элементов по 2.

В этой задаче мы будем рассматривать все k-элементные сочетания с повторениями множества из n чисел от 1 до n. Естественно, что все эти сочетания можно упорядочить лексикографически как вектора, если считать порядок чисел в сочетании неубывающим. Скажем, при $n = 3$ и $k = 2$ список упорядоченных сочетаний с повторениями будет выглядеть так: $(1,1)$, $(1,2)$, $(1,3)$, $(2,2)$, $(2,3)$, $(3,3)$.

В этой задаче требуется найти сочетание с повторениями по лексикографическому номеру (нумерация вёдется с нуля).

Формат входного файла

Три числа n, k и x ($1 \leqslant k \leqslant n \leqslant 30$), x задаёт номер существующего сочетания с повторениями.

Формат выходного файла

Выведите k чисел, задающих требуемое сочетание с повторениями.

Примеры

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

Задача I. Номер по скобочной последовательности

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

Правильной скобочной последовательностью из $2n$ скобок называется такая последовательность, которая может встречаться в некотором арифметическом выражении. Например, `()()()` и `(())()` являются правильными скобочными последовательностями, а `((()))` и `(()))(` — нет.

Все скобочные последовательности можно упорядочить в лексикографическом порядке, считая, что '(' меньше, чем ')'. Скажем, при $n = 3$ список упорядоченных правильных скобочных последовательностей будет выглядеть так: `((()))`, `(()())`, `(())()`, `()(())`, `()()()`.

В этой задаче требуется найти лексикографический номер по правильной скобочной последовательности (нумерация ведётся с нуля).

Формат входного файла

В первой строке дано число n ($1 \leqslant n \leqslant 30$). Во второй строке дана правильная скобочная последовательность из $2n$ скобок.

Формат выходного файла

Выведите номер правильной скобочной последовательности.

Пример

parenthr.in	parenthr.out
3 (()())	1
3 ()()()	4

Задача J. Скобочная последовательность из двух типов скобок по номеру

Имя входного файла:	parenth2.in
Имя выходного файла:	parenth2.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Правильной скобочной последовательностью из двух типов скобок из $2n$ скобок называется такая последовательность

круглых и квадратных скобок, которая может встречаться в некотором арифметическом выражении. Например, `()[]()` и `(())()` являются правильными скобочными последовательностями из двух типов скобок, а `(((()))]` и `([])][` — нет.

Все правильные скобочные последовательности из двух типов скобок можно упорядочить в лексикографическом порядке, считая, что порядок скобок соответствует их кодам символов: '(' < ')' < '[' < ']'. Скажем, при $n = 2$ список упорядоченных правильных скобочных последовательностей из двух типов скобок будет выглядеть так: `(())`, `()()`, `()[]`, `([])`, `[()]`, `[[]]`, `[]()`, `[][]`.

В этой задаче требуется найти правильную скобочную последовательность из двух типов скобок по лексикографическому номеру (нумерация ведётся с нуля).

Формат входного файла

Два числа n и x ($1 \leqslant n \leqslant 20$), x задаёт номер существующей правильной скобочной последовательности из двух типов скобок.

Формат выходного файла

Выведите строку из $2n$ круглых и квадратных скобок, задающих требуемую правильную скобочную последовательность из двух типов скобок.

Пример

parenth2.in	parenth2.out
2 1	`()()`
2 4	`[()]`

Задача K. Перестановка букв строки по номеру

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

Перестановками букв строки S называются все различные строки, получающиеся из неё перестановкой букв.

Естественно, что все эти перестановки букв можно упорядочить лексикографически как строки. Скажем, для строки `abba` список упорядоченных перестановок букв будет выглядеть так: `aabb`, `abab`, `abba`, `baab`, `baba`, `bbaa`.

В этой задаче требуется найти перестановку букв строки по лексикографическому номеру (нумерация ведётся с нуля).

Формат входного файла

В первой строке дана строка S, состоящая не более чем из 20 маленьких букв латинского алфавита. Во второй строке задано число x, которое задаёт номер существующей перестановки букв.

Формат выходного файла

Выведите требуемую перестановку букв строки S.

Пример

string.in	string.out
abba 1	abab
abba 4	baba

Задача L. Перестановка

Имя входного файла:	`perm.in`
Имя выходного файла:	`perm.out`
Ограничение по времени:	0.3 секунды
Ограничение по памяти:	256 мебибайт

Постройте перестановку из $1 \leqslant N \leqslant 1000$ элементов по её лексикографическому номеру $1 \leqslant K \leqslant N!$.

Формат входного файла

В единственной строке входного файла содержатся два числа N и K.

Формат выходного файла

Выведите в единственной строке выходного файла N различных чисел $x_1, \ldots, x_N$, в пределах от 1 до N, задающих требуемую перестановку. Числа разделяются пробелами.

Пример

perm.in	perm.out
3 2	1 3 2

Задача M. Следующая перестановка

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

Рассматривается множество всех перестановок длины n, расположенных в лексикографическом порядке. Требуется по заданной перестановке найти следующую в этом множестве, или вывести, что такой перестановки не существует.

Формат входного файла

Число $1 \leqslant n \leqslant 50000$, и n чисел a_i, задающих перестановку, $1 \leqslant a_i \leqslant n$, $a_i \neq a_j$ при $i \neq j$.

Формат выходного файла

n чисел, задающих следующую перестановку, или слово `IMPOSSIBLE`, если следующей перестановки не существует.

Пример

nextperm.in	nextperm.out
2 1 2	2 1
2 2 1	IMPOSSIBLE
3 2 1 3	2 3 1
10 1 7 2 3 9 10 8 6 5 4	1 7 2 3 10 4 5 6 8 9

Задача N. Следующий элемент

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

В НИИ НАНО (Научно-Исследовательском Институте Неудержимых Амбиций Научного Общества) недавно смастерили действительно уникальное устройство. По замыслу автора, по заданному числу n, а также по разбиению этого числа на слагаемые это устройство выдаёт лексикографически следующее разбиение, а если разбиение было последним — то первое разбиение числа $n + 1$.

Как известно, одним из тестов всех новых НАНО-устройств является крэш-тест. И оказалось, что после падения с 10-го этажа устройство стало работать как-то странно. К примеру, в 10:02 на разбиение 3 2 оно выдало разбиение 5, а в 10:03 на разбиение 5 3 3 оно выдало разбиение 6 5. Автор предположил, что устройство теперь рассматривает в качестве следующих только разбиения, которые не содержат слагаемых, меньших некоторого числа m — например, это может быть количество минут в данный момент времени. В этот же миг — очень кстати! — институт посетил его знакомый учёный Вася.

Как водится, Васе поручили проверить эту гипотезу. От него требуется написать программу, которая бы симулировала ожидаемое поведение устройства. Формально, рассмотрим

бесконечную последовательность $S(m)$ разбиений на слагаемые, которая получается конкатенацией последовательностей $P(m)$, $P(m+1)$, $P(m+2)$, ..., где $P(t)$ — это лексикографически упорядоченная последовательность тех разбиений t на слагаемые, которые не содержат слагаемых, меньших m. Разбиением считается $t = a_1 + a_2 + \ldots + a_k$, где $k \geqslant 1$ и $a_1 \geqslant a_2 \geqslant \ldots \geqslant a_k$. Лексикографический порядок определяется так: если в $P(t)$ разбиение a встречается раньше разбиения b, то для первой позиции p такой, что a_p и b_p не совпадают, верно $a_p < b_p$. Например, начало последовательности $S(2)$ таково: 2, 3, 2 2, 4, 3 2, 5, 2 2 2, 3 3, 4 2, 6, ...

По своему обыкновению, Вася перепоручает эту задачу вам. По данному числу m, а также элементу последовательности $S(m)$, найдите следующий элемент этой последовательности.

Формат входного файла

В первой строке ввода заданы два целых числа k и m ($1 \leqslant k \leqslant 50\,000$, $1 \leqslant m \leqslant 50\,000$). Во второй строке задано k целых чисел a_i — элемент последовательности $S(m)$ ($m \leqslant a_i \leqslant 10^6$). Гарантируется, что $a_1 \geqslant a_2 \geqslant \ldots \geqslant a_k$.

Формат выходного файла

В первой строке выведите k' — количество слагаемых в следующем элементе $S(m)$. Во второй строке выведите k' чисел — слагаемые, составляющие следующий элемент $S(m)$, в невозрастающем порядке. Гарантируется, что k' не превосходит 50 000.

Примеры

next.in	next.out
2 2 3 2	1 5
3 3 5 3 3	2 6 5
1 2 5	3 2 2 2