Задача A. Генератор

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

Даны два натуральных числа N и K. Требуется вывести все цепочки x_1, x_2, ..., x_N такие, что x_i — натуральное число и $1 \leqslant x_i \leqslant K$.

Формат входного файла

Вводятся два натуральных числа N и K ($N, K \leqslant 6$).

Формат выходного файла

Выведите все требуемые цепочки в произвольном порядке — по одной на строке. Никакая цепочка не должна встречаться более одного раза.

Примеры

generator.in	generator.out
2 3	1 1 1 2 1 3 2 1 2 2 2 3 3 1 3 2 3 3

Задача B. Двоичные строки

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

По данным числам n и k выведите все строки из нулей и единиц длины n, содержащие ровно k единиц, в лексикографическом порядке.

Формат входного файла

Во входном файле даны два целых числа числа — n и k ($0 \leqslant k \leqslant n \leqslant 100$).

Формат выходного файла

Необходимо вывести все строки из нулей и единиц длины n, содержащие ровно k единиц, в лексикографическом порядке. Гарантируется, что размер ответа не превышает 10 мебибайт.

Примеры

binary.in	binary.out
4 2	0011 0101 0110 1001 1010 1100

Задача C. Перестановки

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

Дана строка, состоящая из m символов. Требуется вывести все перестановки символов данной строки.

Формат входного файла

В первой строке файла находится исходная строка, состоящая только из букв латинского алфавита и цифр. Длина строки удовлетворяет условию $2 \leqslant m \leqslant 8$.

Формат выходного файла

Требуется вывести в каждой строке файла по одной перестановке. Перестановки можно выводить в любом порядке. Повторений и строк, не являющихся перестановками исходной, быть не должно.

Примеры

permutations.in	permutations.out
AB	AB BA
122	122 212 221

Задача D. Правильные скобочные последовательности

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

Правильная скобочная последовательность — это такая последовательность, которая могла быть получена из арифметического выражения удалением чисел, констант, переменных и знаков арифметических действий.

Более строго, существует следующее определение правильной скобочной последовательности:

1. пустая строка — правильная скобочная последовательность;
2. правильная скобочная последовательность, взятая в скобки — правильная скобочная последовательность;
3. правильная скобочная последовательность, к которой приписана слева или справа правильная скобочная последовательность — тоже правильная скобочная последовательность;
4. строки, не подходящие под правила 1, 2 и 3, правильными скобочными последовательностями не являются.

В данной задаче от вас требуется простое: получить список всех правильных скобочных последовательностей, состоящих из n открывающих и n закрывающих скобок.

Формат входного файла

Во входном файле дано одно целое число — n ($1 \leqslant n \leqslant 12$).

Формат выходного файла

Необходимо вывести в произвольном порядке все правильные скобочные последовательности из n пар скобок.

Примеры

parentheses.in	parentheses.out
2	(()) ()()

Задача E. Количество циклов

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

Формально, *путь* в графе — это чередующаяся последовательность вершин и рёбер u_1, e_1, u_2, e_2, $u_3, \ldots, u_k$, начинающаяся и заканчивающаяся вершиной и такая, что любые соседние вершина и ребро в ней инцидентны.

Цикл — это путь, начальная и конечная вершины которого совпадают. В цикле должно быть хотя бы одно ребро.

Простой путь отличается от обычного пути тем, что в нём не может быть повторяющихся вершин.

Простой цикл — это цикл, в котором нет повторяющихся вершин и рёбер.

Дан неориентированный граф. Посчитайте, сколько в нём различных простых циклов. Заметим, что циклы считаются одинаковыми, если они обходят одно и то же множество вершин в одном и том же порядке, возможно, начиная при этом из другой вершины, или если порядок обхода противоположный. Например, циклы с порядком обхода вершин $1, 2, 3, 1$, $2, 3, 1, 2$ и $1, 3, 2, 1$ считаются одинаковыми, а циклы $1, 2, 3, 4, 1$ и $1, 3, 4, 2, 1$ — нет, поскольку порядок обхода вершин различен.

Формат входного файла

В первой строке входного файла заданы числа N и M через пробел — количество вершин и рёбер в графе, соответственно ($1 \leqslant N \leqslant 10$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leqslant u_i, v_i \leqslant N$, $u_i \neq v_i$); каждая такая строка означает, что в графе существует ребро между вершинами u_i и v_i. В графе нет кратных рёбер.

Формат выходного файла

Выведите одно число — количество простых циклов в заданном графе.

Примеры

numcycle.in	numcycle.out
3 2 1 2 2 3	0
4 5 1 2 2 3 3 4 4 1 1 3	3

Задача F. Различные разбиения

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

Найдите количество различных разбиений натурального числа n на натуральные слагаемые таких, что для любых двух различных чисел $a \neq b$, входящих в разбиение, верно, что количества чисел a и b в разбиении различны. Разбиения, отличающиеся только порядком слагаемых, различными не считаются.

Например, если $n = 4$, то из пяти возможных разбиений этому условию удовлетворяют все, кроме разбиения на слагаемые 1 и 3: в этом разбиении количество единиц равно количеству троек.

4	$=$	$1+1+1+1$	4 единицы
4	$=$	$1+1+2$	3 единицы, 1 тройка
4	$=$	$1+3$	1 единица и 1 тройка!
4	$=$	$2+2$	2 двойки
4	$=$	4	1 четвёрка

Формат входного файла

В первой строке входного файла записано натуральное число n ($1 \leq n \leq 100$).

Формат выходного файла

В первой строке выходного файла выведите количество разбиений числа n, удовлетворяющих заданным ограничениям.

Примеры

numdiff.in	numdiff.out
4	4
6	7