Задача A. Строка Фибоначчи

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Строка Фибоначчи — это строка из нулей и единиц, в которой не встречается двух идущих подряд единиц.

Даны числа n и k. Нужно вывести строку Фибоначчи, которая состоит из n цифр и является k-й в лексикографическом порядке из таких строк.

Формат входных данных

В первой строке записаны целые числа n и k через пробел ($0 \leqslant n \leqslant 44$, $0 \leqslant k \leqslant 2 \cdot 10^9$). Гарантируется, что k-я строка из n символов существует. Строки нумеруются с нуля.

Формат выходных данных

Выведите лексикографически k-ю строку Фибоначчи длины n.

Примеры

стандартный ввод	*стандартный вывод*
3 0	000
3 1	001
3 2	010
3 3	100
3 4	101

Задача B. Перестановка по номеру

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Перестановкой из n элементов называется некоторый набор чисел от 1 до n, записанных в произвольном порядке, в котором каждое число встречается ровно один раз.

Говорят, что одна перестановка *лексикографически меньше* другой, если первые k чисел в них совпадают, а $(k+1)$-е число в первой меньше, чем во второй.

Набор перестановок называется *лексикографически упорядоченным*, если в нём каждая перестановка лексикографически меньше, чем следующая.

Ясно, что из всех n-элементных перестановок можно единственным образом построить лексикографически упорядоченный набор. Например, для $n = 3$ таким набором будет ((1 2 3), (1 3 2), (2 1 3), (2 3 1), (3 1 2), (3 2 1)).

Лексикографическим номером перестановки из n элементов называется ее номер в лексикографически упорядоченном наборе всех n-элементных перестановок. Например, перестановка (2 3 1) будет иметь лексикографический номер 4.

Вам требуется написать программу, которая будет находить перестановку по числу n ($1 \leqslant n \leqslant 20$) и её лексикографическому номеру k.

Формат входных данных

В первой строке заданы два числа: n и k. Число k не превосходит 10^{18} по модулю.

Формат выходных данных

Вам необходимо выдать саму перестановку — n чисел, или сообщение «`INVALID number.`» (без кавычек), если перестановки с таким номером не существует.

Примеры

стандартный ввод	*стандартный вывод*
3 1	1 2 3
3 2	1 3 2
3 4	2 3 1
3 7	INVALID number.

Задача C. Номер по перестановке

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Вам требуется написать программу, которая будет находить по перестановке её лексикографический номер среди всех перестановок из n различных чисел от 1 до n.

Формат входных данных

В первой строке задано число n ($1 \leqslant n \leqslant 20$). Во второй строке заданы n чисел, образующих перестановку.

Формат выходных данных

Вам необходимо выдать одно число — лексикографический номер перестановки (нумерация начинается с единицы).

Примеры

стандартный ввод	*стандартный вывод*
3 1 2 3	1
3 1 3 2	2
3 2 3 1	4

Задача D. Размещение по номеру

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 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 размещение по лексикографическому номеру (нумерация ведётся с нуля).

Формат входных данных

Три числа n, k и x ($1 \leqslant k \leqslant n \leqslant 20$), x задаёт номер существующего размещения.

Формат выходных данных

Выведите k чисел, задающих требуемое размещение.

Примеры

стандартный ввод	*стандартный вывод*
3 2 1	1 3
4 2 5	2 4

Задача E. Номер по размещению

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 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 входных данных

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 выходных данных

Выведите номер размещения.

Примеры

стандартный ввод	*стандартный вывод*
3 2 1 3	1
4 2 2 4	5

Задача F. Скобочная последовательность по номеру

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 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 правильную скобочную последовательность по лексикографическому номеру (нумерация ведётся с нуля).

Формат входных данных

Два числа n и x ($1 \leqslant n \leqslant 30$), x задаёт номер существующей правильной скобочной последовательности.

Формат выходных данных

Выведите строку из $2n$ скобок, задающих требуемую правильную скобочную последовательность.

Примеры

стандартный ввод	*стандартный вывод*
3 1	(()())
3 4	()()()

Задача G. Номер по скобочной последовательности

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 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 входных данных

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

Формат выходных данных

Выведите номер правильной скобочной последовательности.

Примеры

стандартный ввод	*стандартный вывод*
3 `(()())`	1
3 `()()()`	4

Задача H. Три типа скобок

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Определим по индукции множество $\mathcal{T}$ *правильных скобочных последовательностей из трёх типов скобок*:

- $\varepsilon \in \mathcal{T}$ (пустая строка)
- $A \in \mathcal{T} \Rightarrow$ `(`A`)` $\in \mathcal{T}$
- $A \in \mathcal{T} \Rightarrow$ `[`A`]` $\in \mathcal{T}$
- $A \in \mathcal{T} \Rightarrow$ `{`A`}` $\in \mathcal{T}$
- $A \in \mathcal{T}, B \in \mathcal{T} \Rightarrow AB \in \mathcal{T}$

Пусть теперь $\mathcal{T}_n$ — это множество правильных скобочных последовательностей из $2n$ символов — n открывающих и n закрывающих скобок.

Упорядочим элементы множества $\mathcal{T}_n$ лексикографически с некоторым порядком символов «`(`», «`)`», «`[`», «`]`», «`{`» и «`}`».

По данным числам n и p, а также порядку, заданному на скобках, найдите p-ый в этом порядке элемент множества $\mathcal{T}_n$.

Формат входных данных

В первой строке заданы через пробел два целых числа n и p ($0 \leqslant n \leqslant 20$, $0 \leqslant p \leqslant 9 \cdot 10^{18}$). Скобочные последовательности нумеруются с нуля.

Во второй строке записаны шесть символов — «`(`», «`)`», «`[`», «`]`», «`{`» и «`}`» — в некотором порядке. Их порядок задаёт лексикографический порядок на множестве $\mathcal{T}_n$.

Формат выходных данных

В первой строке выведите $2n$ символов без пробелов — p-ю правильную скобочную последовательность длины $2n$ из трёх типов скобок.

Если для данного n не существует p-я правильная скобочная последовательность, выведите в первой строке «`N/A`».

Примеры

стандартный ввод	*стандартный вывод*
1 0 ()}[{]	()
1 1 ()}[{]	[]
1 2 ()}[{]	{}
1 3 ()}[{]	N/A

Задача I. Волшебный замок

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Доблестный воин Феофорд недавно приобрёл у гномов волшебный замóк для своего сундука. *Волшебный замок* имеет форму круглого диска. На границе лицевой стороны диска в вершинах правильного многоугольника расположены $2n$ шариков, пронумерованных числами от 1 до $2n$ в порядке следования по часовой стрелке. Эти шарики разбиты на n пар так, что, если соединить каждую пару отрезком, никакие два отрезка не будут пересекаться. Открыть замок можно, начертив на поверхности диска n линий, соединяющих парные шарики. Линии можно проводить в любом порядке.

К сожалению, Феофорд забыл, какие пары шариков нужно соединять, и теперь не может открыть свой сундук! Всё, что он помнит — это серийный номер своего волшебного замка. К счастью, зайдя на сайт технической поддержки, Феофорд узнал, что этой информации достаточно, чтобы открыть замок. Однако что и как нужно сделать, Феофорд не разобрался. Вот инструкция, которую он нашёл на гномьем сайте:

> *Кодом* для замка является последовательность из $2n$ чисел: какая точка состоит в паре с первой, какая — со второй, ..., какая — с $(2n)$-й. Очевидно, по коду легко восстановить, какие линии нужно провести.
>
> Выпишем всевозможные коды к замкам с $2n$ шариками и упорядочим их *лексикографически* как последовательности из $2n$ чисел. Одна последовательность будет выписана раньше другой, если в первой позиции, где они различаются, элемент первой последовательности меньше элемента второй.
>
> *Серийный номер* замка получается просто как номер кода этого замка в лексикографическом порядке. Коды нумеруются, начиная с единицы.

Помогите Феофорду по серийному номеру замка восстановить код для него.

Формат входных данных

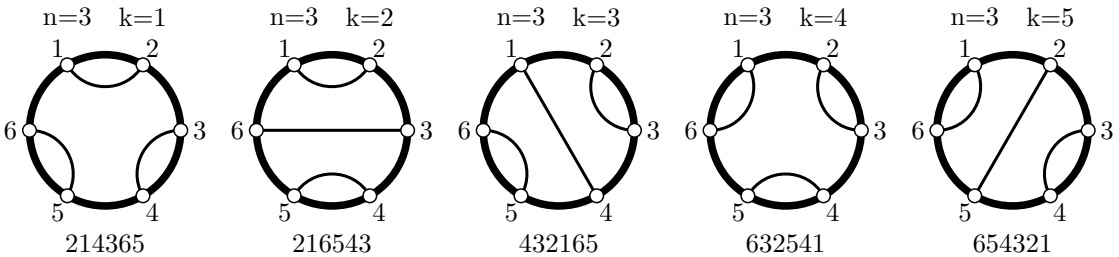
Первая строка ввода содержит одно целое число t — количество тестов ($1 \leqslant t \leqslant 1000$). Каждая из t следующих строк содержит два целых числа n и k — количество пар шариков и серийный номер замка ($1 \leqslant n \leqslant 35$, $1 \leqslant k \leqslant 4 \cdot 10^{18}$).

Формат выходных данных

В ответ на каждый тест выведите строку, содержащую $2n$ целых чисел, разделённых пробелами — k-й в лексикографическом порядке код замка с $2n$ шариками.

Пример

стандартный ввод	*стандартный вывод*
5 3 1 3 2 3 3 3 4 3 5	2 1 4 3 6 5 2 1 6 5 4 3 4 3 2 1 6 5 6 3 2 5 4 1 6 5 4 3 2 1



Задача J. Биекция

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Рассмотрим пути на плоскости из $(0, 0)$ в (n, n), состоящие из единичных шагов вправо («R») и вверх («U»). Известно, что количество различных таких путей равно биномиальному коэффициенту

$$\text{choose}(2n, n) = \frac{(2n)!}{n! \cdot n!}.$$

Например, при $n = 2$ существует шесть таких путей: «RRUU», «RURU», «RUUR», «URRU», «URUR», «UURR».

Строка U называется правильной скобочной последовательностью, если это пустая строка, или строка вида «(V)», или же конкатенация двух строк вида «VW», где V и W — правильные скобочные последовательности. Рассмотрим правильные скобочные последовательности из n пар скобок. Известно, что количество различных таких последовательностей равно числу Каталана, которое можно вычислить, в частности, так:

$$C_n = \frac{1}{n+1} \cdot \text{choose}(2n, n).$$

Например, при $n = 2$ существует две таких последовательности: «(())», «()()».

Постройте любую биекцию, отражающую этот факт. А именно, зная путь из n шагов вправо и n шагов вверх, постройте правильную скобочную последовательность из n пар скобок, а также запомните целое число k от 0 до n включительно. Затем, получив эту последовательность и это число, восстановите исходный путь.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза.

При первом запуске решение кодирует путь. В первой строке записано слово «path». Вторая строка содержит целое число n — половину длины пути ($1 \leqslant n \leqslant 300$). В третьей строке записан путь из $2n$ шагов: n букв «R» и n букв «U» в каком-то порядке.

В первой строке выведите любую правильную скобочную последовательность из n символов «(» и n символов «)». Во второй строке выведите любое целое число k ($0 \leqslant k \leqslant n$).

При втором запуске решение восстанавливает путь. В первой строке записано слово «brackets». Вторая строка содержит целое число n, то же, что и при первом запуске — половину длины скобочной последовательности ($1 \leqslant n \leqslant 300$). В третьей строке записана правильная скобочная последовательность из n символов «(» и n символов «)». В четвёртой строке записано целое число k ($0 \leqslant k \leqslant n$). Последовательность и число — ровно те, которые решение вывело при первом запуске.

В первой строке выведите восстановленный исходный путь: n букв «R» и n букв «U» в том же порядке, что и во входных данных при первом запуске.

При всех запусках каждая строка входных данных, включая последнюю, завершается переводом строки.

Примеры

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	***стандартный вывод***
path 2 RRUU	(()) 0
brackets 2 (()) 0	RRUU

Далее показаны два запуска какого-то решения на втором тесте.

стандартный ввод	***стандартный вывод***
path 3 RUURRU	(()()) 3
brackets 3 (()()) 3	RUURRU