Задача A. Различные слагаемые

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

Задано целое положительное число n. Сколько существует способов представить его в виде суммы различных целых положительных слагаемых? Способы, отличающиеся только порядком слагаемых, считаются одинаковыми.

Поскольку ответ может быть очень большим, найдите остаток от деления искомого количества на простое число $998\,244\,353$.

Формат входных данных

В первой строке записано целое число n ($1 \leqslant n \leqslant 500\,000$).

Формат выходных данных

Выведите одно целое число: количество способов представить n в виде суммы различных целых положительных слагаемых, взятое по модулю $998\,244\,353$.

Примеры

стандартный ввод	стандартный вывод	пояснение
5	3	5 = 1 + 4 5 = 2 + 3 5 = 5
10	10	10 = 1 + 2 + 3 + 4 10 = 1 + 2 + 7 10 = 1 + 3 + 6 10 = 1 + 4 + 5 10 = 1 + 9 10 = 2 + 3 + 5 10 = 2 + 8 10 = 3 + 7 10 = 4 + 6 10 = 10

Задача B. Разрезание многоугольника

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

Определите количество способов разрезать выпуклый многоугольник с n сторонами на k частей. Разрезы разрешено проводить только от вершины к вершине, таким образом, что многоугольник делится ровно на две части. Вершины считаются пронумерованными различными числами, порядок разрезов не важен.

Например, квадрат можно разрезать тремя способами. Два из них — произвести разрез по какой-либо из диагоналей (таким образом получается две части), и один — вообще не производить разрезов (получается одна часть).

Так как количество способов может быть очень большим, то нужно вывести ответ по модулю 10^9 (то есть последние 9 цифр).

Формат входных данных

В единственной строке заданы два целых числа n и k ($3 \leqslant n \leqslant 100$, $1 \leqslant k \leqslant 100$).

Формат выходных данных

Выведите количество способов по модулю 10^9. Если многоугольник невозможно разрезать на k частей, выведите -1.

Примеры

стандартный ввод	стандартный вывод
4 2	2
100 1	1
6 4	14
31 20	956146480
3 4	-1

Задача C. 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 D. 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 E. 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 F. 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 G. 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

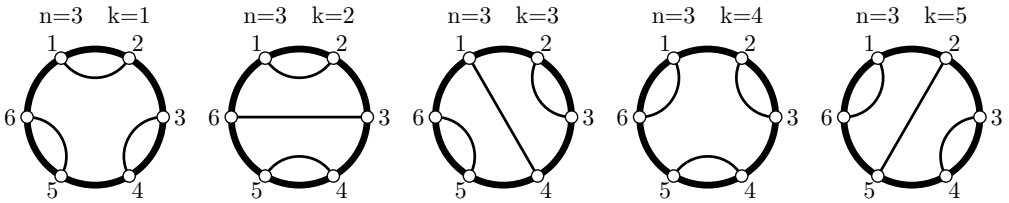
Первая строка ввода содержит одно целое число t — количество тестов ($1 \leqslant t \leqslant 1000$). Каждая из t следующих строк содержит два целых числа n и k — количество пар шариков и серийный номер замка ($1 \leqslant n \leqslant 35$, $1 \leqslant k \leqslant 4 \cdot 10^{18}$).

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



214365 216543 432165 632541 654321

Задача H. Чётные и нечётные сочетания

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

Сочетанием из n элементов по k будем называть k-элементное подмножество n-элементного множества $\{1, 2, \ldots, n\}$. Чтобы записать сочетание, перечислим его элементы в порядке возрастания. Например, сочетания из 3 элементов по 2 выглядят так: $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$.

Будем называть сочетание *чётным*, если количество элементов в нём — чётное число, и *нечётным* в противном случае. Зафиксируем $n > 0$ и рассмотрим два множества: A_n, множество всех чётных сочетаний из n элементов, и B_n, множество всех нечётных сочетаний из n элементов. Можно доказать, что A_n и B_n содержат одинаковое количество сочетаний.

Для каждого $n = 1, 2, \ldots, 50$ задача такова. Постройте любую биекцию (взаимно однозначное соответствие) между множествами A_n и B_n. После этого по данному элементу одного из этих множеств выводите соответствующий ему элемент другого множества.

Протокол взаимодействия

Для проверки того, что вы действительно построили биекцию, в этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске в первой строке записано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Далее следуют их описания.

Каждый тестовый случай задаёт сочетание и занимает две строки. В первой из них записаны через пробел два целых числа n и k ($1 \leqslant n \leqslant 50$, $0 \leqslant k \leqslant n$). Во второй записаны через пробел k целых чисел $a_1, a_2, \ldots, a_k$ — элементы сочетания ($1 \leqslant a_1 < a_2 < \ldots < a_k \leqslant n$). Если $k = 0$, то вторая строка пуста.

В ответ на каждый тестовый случай выведите сочетание из другого множества, соответствующее заданному. Формат сочетания в выводе — точно такой же, как во вводе. У выведенного сочетания n должно совпадать с заданным, а k должно иметь другую чётность. Других ограничений на выбор соответствия нет.

Второй запуск

При втором запуске формат ввода точно такой же, как при первом запуске. Но в каждом тестовом случае дано не исходное сочетание, а то, которое было выведено при первом запуске.

В ответ на каждый тестовый случай, как и при первом запуске, выведите сочетание из другого множества, соответствующее заданному. Формат сочетания в выводе — точно такой же, как во вводе. Поскольку соответствие должно быть взаимно однозначным, выведенное при втором запуске сочетание должно совпадать с тем, которое было дано при первом запуске. Это и будет проверять программа жюри.

Пример

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*
6 3 0 2 1 1 3 3 1 2 3 3 1 1 3 1 2 3 1 3	3 3 1 2 3 2 2 1 2 3 0 3 2 2 3 3 2 1 3 3 2 1 2

стандартный ввод	*стандартный вывод*
6 3 3 1 2 3 2 2 1 2 3 0 3 2 2 3 3 2 1 3 3 2 1 2	3 0 2 1 1 3 3 1 2 3 3 1 1 3 1 2 3 1 3

Задача I. 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

Задача J. Скобочно-палочные последовательности

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

Определим множество *правильных скобочно-палочных последовательностей* R рекурсивно. Это множество строк, которые можно получить, применяя только следующие правила:

- $\varepsilon \in R$ (пустая строка)
- $A, B \in R \Rightarrow AB \in R$ (конкатенация)
- $A, B \in R \Rightarrow$ `(`A`|`B`)` $\in R$

Например, последовательности, содержащие две тройки «`(|)`», выглядят так: «`(()|)|)`», «`(|(|))`», «`(|)(|)`».

Придумайте, как сопоставить правильным скобочно-палочным последовательностям заданной длины целые числа, и реализуйте это сопоставление.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске решение кодирует скобочно-палочные последовательности целыми числами. В первой строке записано слово «`encode`». Вторая строка содержит целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Каждый тестовый случай занимает две строки: первая содержит целое число n — количество троек «`(|)`» в последовательности ($1 \leqslant n \leqslant 25$), а вторая — $3n$ символов без пробелов, которые образуют правильную скобочно-палочную последовательность из n троек.

Выведите t строк, по одной на каждый тестовый случай. В i-й строке выведите целое число x_i, которым вы хотите закодировать i-ю последовательность из входных данных ($0 \leqslant x_i \leqslant 2 \cdot 10^{18}$).

Второй запуск

При втором запуске решение декодирует скобочно-палочные последовательности из целых чисел. В первой строке записано слово «`decode`». Вторая строка содержит целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Каждый тестовый случай занимает две строки: первая содержит целое число n — количество троек «`(|)`» в последовательности ($1 \leqslant n \leqslant 25$), а вторая — целое число, выведенное в этом тестовом случае при первом запуске.

Выведите t строк, по одной на каждой тестовый случай. В i-й строке выведите скобочно-палочную последовательность из i-го тестового случая.

Пример

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. Как видно, это решение кодирует символы цифрами 1, 2 и 3, после чего выводит в качестве числа получившуюся строку из цифр. Увы, для больших n строки окажутся слишком длинными.

стандартный ввод	*стандартный вывод*
encode 3 1 (|) 4 ((((|)|)|)|) 5 (|(|))((|(|))|)	123 111123232323 121233112123323

стандартный ввод	*стандартный вывод*
decode 3 1 123 4 111123232323 5 121233112123323	(|) ((((|)|)|)|) (|(|))((|(|))|)