Задача A. Уравнение с НОК

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Даны два натуральных числа a и b. Требуется найти количество различных натуральных x, удовлетворяющих равенству $\text{НОК}(x, a) = b$.

Формат входных данных

В единственной строке через пробел записаны два целых числа числа a и b ($1 \leqslant a, b \leqslant 10^{12}$).

Формат выходных данных

Выведите единственное число — количество различных натуральных x, удовлетворяющих равенству.

Примеры

стандартный ввод	стандартный вывод
3 15	2
22 1	0

Замечание

Напомним, что $\text{НОК}(n, m)$ — это такое минимальное положительное число k, что k делится без остатка и на n, и на m.

В первом примере подходящие x — это 5 и 15.

Во втором примере подходящих x не существует.

Задача B. Странный Порядок

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Как часто вам приходится упорядочивать какие-либо объекты? Сортировать числа? Или, может быть, строки? В этой задаче вам предстоит упорядочить числа согласно довольно необычному критерию.

У вас имеется N целых положительных чисел $a_1, a_2, \ldots, a_N$. От вас требуется переупорядочить данные числа таким образом, чтобы для любых a_i и a_{i+1} были выполнены условия:

1. Пусть $c_1, c_2, \ldots, c_k$ — всевозможные положительные делители числа a_i, причем $c_i < c_{i+1}$. Аналогично $d_1, d_2, \ldots, d_l$ — делители числа a_{i+1}. Должно быть верно, что либо $k < l$, либо $k = l$, и при этом для любого j верно, что $c_j \leqslant d_j$.

2. a_i `AND` $a_{i+1} = a_i$, где `AND` — операция побитового «И».

Формат входных данных

В первой строке записано целое число N ($1 \leqslant N \leqslant 10^6$) — количество элементов, которые необходимо упорядочить.

Во второй строке через пробел записаны N положительных целых чисел $a_1, a_2, \ldots, a_N$ ($1 \leqslant a_i \leqslant 10^{12}$).

Формат выходных данных

В первой строке выведите «`Yes`» (без кавычек), если упорядочить числа требуемым образом возможно, либо «`No`» (без кавычек), если необходимого порядка чисел не существует.

В случае, если упорядочить числа возможно, во второй строке выведите N исходных чисел в нужном порядке.

Примеры

стандартный ввод	стандартный вывод
2 2 6	Yes 2 6
3 1 2 3	No
3 7 1 3	Yes 1 3 7

Задача C. Теория Игр

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Костя отыскал у себя дома N кубиков и принялся играть в одну увлекательную игру: он захотел воспользоваться находкой и построить максимально высокую башню!

У Кости есть большой стол, на котором изначально кубиков нет. Строительство башни состоит из последовательности действий Кости. Во время каждого действия Костя берет очередной кубик и ставит его сверху на уже построенную башню, либо на стол, если ранее на столе кубиков не было. После каждого действия Костя выписывает на лист количество кубиков в получившейся башне. Когда у Кости кубиков не останется, он закончит игру.

Его другу Мише такая игра показалась скучной, и он решил в неё вмешаться. После того, как Костя совершит очередное действие и запишет на лист количество кубиков в новой башне, Миша будет делать одно из двух действий:

- Не вмешиваться в игру и позволить Косте продолжать строительство башни.
- Разрушить башню, которая на данный момент находится на столе и забрать все кубики, из которых состояла башня, себе. Если Миша забрал себе некоторые кубики, он уже ни за что не отдаст их Косте.

Если Миша вмешается в игру после очередного действия Кости, чтобы сильно не расстраиваться и не обижаться на друга, Костя запишет себе на лист поощрительное число C. После этого он продолжит свою игру, начав строить башню заново уже из оставшихся кубиков.

Миша не любит большие числа и хочет, чтобы сумма всех записей, сделанных Костей во время игры была как можно меньше.

Помогите Мише найти сумму чисел, записанных Костей, в конце игры, если он будет действовать оптимально.

Формат входных данных

В единственной строке через пробел записаны два целых числа N и C ($1 \leqslant N, C \leqslant 10^9$) — количество кубиков у Кости и поощрительное число, соответственно.

Формат выходных данных

Выведите одно целое число — минимальную сумму чисел, записанных Костей на лист.

Примеры

стандартный ввод	стандартный вывод
3 3	6
5 9	15
1000000 1	1999999

Замечание

В первом примере оптимальной стратегией Миши является не вмешиваться в игру и позволить Косте построить башню, состоящую из трех кубиков. Тогда сумма записанных чисел будет равна: $1 + 2 + 3 = 6$.

Во втором примере также следует не мешать Косте и позволить построить башню из пяти кубиков.

В третьем примере оптимальным является разрушать башню после каждого действия Кости, кроме последнего. Таким образом, сумма записанных чисел будет равна $1\,000\,000 + 1 \cdot 999\,999$.

Задача D. Стековая Машина Возвращается

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Ровно год назад на Пятой Липецкой командной олимпиаде школьников по программированию участникам была предложена задача, в которой нужно было вычислить сумму элементов массива, используя довольно необычный язык программирования и вычислительную машину. Вы можете вспомнить условие той самой задачи ниже.

Не так давно вы устроились на работу в известную компанию CodeHorses, которая занимается проведением соревнований по программированию. Вы подумали, что будет замечательно, если у участников появится возможность решать задачи по программированию при помощи стековой машины! Поэтому вы немедленно приступили к работе по созданию компилятора языка программирования для данной вычислительной машины.

Вам будет дана программа, написанная на языке стековой машины, исходное состояние стека и ограничение на максимальное время выполнения программы в тактах. От вас требуется определить корректность данной программы и состояние стека после окончания работы программы.

Формально, возможны четыре варианта развития событий:

1. Программа является синтаксически некорректной. В данном случае результатом считается `Compilation error`.
2. Программа является синтаксически корректной, однако завершается с ошибкой (например, `POP` из пустого стека). В этом случае результатом считается `Runtime error`.
3. Программа является синтаксически корректной, однако не успевает завершиться за указанное количество тактов. В этом случае результатом считается `Time limit exceeded`.
4. Программа является синтаксически корректной и завершается без ошибок не более, чем за указанное количество тактов. В этом случае результатом считается `Accepted`. В данном случае необходимо помимо вердикта вывести конечное состояние стека.

Ниже приведены выдержки из задачи прошлого года, описывающие машину, а также допустимые операции и их синтаксис. Если вы хорошо помните задачу, можете продолжать чтение с раздела «`Входные данные`».

У вас есть машина, память которой — это стек, элементы которого являются целыми числами. Количество элементов стека в данный момент мы будем обозначать как sz.

Напомним, что стек — это абстрактная структура данных, хранящая коллекцию элементов, и работающая по принципу `LIFO` (последним пришёл — первым вышел). Стек поддерживает две базовые операции: положить элемент на вершину стека, взять элемент с вершины стека. В качестве аналогии можно представить стопку тарелок: вы не можете оперировать с тарелками не на вершине стопки.

Список доступных инструкций приведен ниже. `RTE` (Runtime Error) обозначает, что выполнение программы завершилось с ошибкой.

- `PUSHZ` — положить на стек 0;
- `POP` — взять со стека x (если $sz < 1$, то `RTE`);
- `SWAP2` — взять со стека x, взять со стека y, положить на стек x, положить на стек y (если $sz < 2$, то `RTE`);
- `SWAP3` — взять со стека x, взять со стека y, взять со стека z, положить на стек y, положить на стек x, положить на стек z (если $sz < 3$, то `RTE`);

- `COPY` — взять со стека x, положить на стек x, положить на стек x (если $sz < 1$, то `RTE`);
- `INC` — взять со стека x, положить на стек $x + 1$ (если $sz < 1$, то `RTE`);
- `DEC` — взять со стека x, положить на стек $x - 1$ (если $sz < 1$, то `RTE`);
- `ADD` — взять со стека x, взять со стека y, положить на стек $x + y$ (если $sz < 2$, то `RTE`);
- `SUB` — взять со стека x, взять со стека y, положить на стек $x - y$ (если $sz < 2$, то `RTE`);

Также доступны условный оператор и цикл while. Чтобы их использовать, сначала нужно научиться задавать какие-то условия:

- `EZ` — `TRUE` если на вершине стека 0 (если $sz < 1$, то `RTE`);
- `GZ` — `TRUE` если на вершине стека положительное число (если $sz < 1$, то `RTE`);
- `HAVE1`, `HAVE2` — `TRUE` если $sz \geqslant k$ (обратите внимание, что $k \in \{1, 2\}$, `HAVE3` не является условием)

Условный оператор:

```
IF cond THEN
BEGIN
    body
END
```

Цикл while:

```
WHILE cond DO
BEGIN
    body
END
```

В обоих случаях `cond` — одно из условий, описанных выше, `body` — тело условного оператора или цикла — последовательность команд, которые будут выполнены, если `cond=TRUE` (один раз в случае условного оператора, или много раз в случае цикла). Циклы и условные операторы могут быть вложенными.

Любой пробел и перевод строки может быть заменён на любое ненулевое количество пробелов и переводов строки, вставлять пробелы и переводы строки внутрь инструкций нельзя.

Каждая инструкция (а также проверки условия для условного оператора и цикла) выполняется 1 такт.

Верхний регистр в названиях инструкций важен.

Формат входных данных

В первой строке записаны два числа n и t ($0 \leqslant n \leqslant 100$, $1 \leqslant t \leqslant 10\,000$) — количество элементов, исходно находящихся в стеке, а также ограничение на время работы в тактах.

Во второй строке через пробел записаны n целых чисел $a_1, a_2, \ldots, a_n$ ($0 \leqslant a_i \leqslant 10^9$) — числа, изначально находящиеся в стеке в данном порядке, причем число a_1 лежит внизу стека, а число a_n — наверху.

В следующих строках находится код программы, написанный на языке стековой машины. Код состоит из символов английского алфавита («`a-z`» и «`A-Z`»), цифр («`0-9`»), пробелов (ASCII-код 32) и символов перевода строки (ASCII-коды 10 и 13).

Гарантируется, что размер каждого теста не превосходит 256 килобайт. Также гарантируется, что программа не является пустой.

Формат выходных данных

В случае, если программа является синтаксически некорректной, в единственной строке выведите «`Compilation error`» (без кавычек).

В случае, если программа является синтаксически корректной, однако завершается с ошибкой, в единственной строке выведите «`Runtime error`» (без кавычек).

В случае, если программа является синтаксически корректной, однако не успевает завершиться за t тактов, в единственной строке выведите «`Time limit exceeded`» (без кавычек).

В противном случае в первой строке выведите «`Accepted`» (без кавычек).

Если программа завершилась корректно, во второй строке выведите число $k \geqslant 0$ — количество чисел, находящихся в стеке после завершения работы программы. В третьей строке выведите k целых чисел $r_1, r_2, \ldots, r_k$ — элементы, находящиеся в стеке, причем число r_1 лежит внизу стека, а число r_k — наверху.

Примеры

стандартный ввод	стандартный вывод

6 100
5 1 2 3 4 5
WHILE HAVE2 DO
BEGIN
    DEC
    ADD
END
	Accepted
	1
	15

6 15
5 1 2 3 4 5
WHILE HAVE2 DO
BEGIN
    DEC
    ADD
END
	Time limit exceeded

6 100
4 8 15 16 23 42
POP
POP
POP
POP
POP
POP
IF EZ THEN
BEGIN
    PUSHZ
END
	Runtime error

2 1000
1 100
WHILE GZ DO
BEGIN
    SWAP2
    COPY
    ADD
    SWAP2
    DEC
END
 	
POP
	Accepted
	1
	126765060022822940149670320537 6

6 100
5 1 2 3 4 5
while HAVE5 DO
BEGIN
    DEC
    SUM
END
	Compilation error

Замечание

Программа из первого примера является решением задачи прошлого года.

Программа из второго примера является такой же, как и в первом примере, однако теперь огра-

ничение на количество тактов равно 15. Так как проверка в цикле также выполняется за один такт, данной программе для завершения потребуется 16 тактов, что не укладывается в ограничение.

В третьем примере происходит проверка `EZ` в пустом стеке, что приводит к ошибке.

В четвертом примере приведена программа, вычисляющая значение 2^{100}.

Программа из пятого примера является синтаксически некорректной по нескольким причинам. Во-первых, ключевое слово «`WHILE`» написано не в правильном регистре. Во-вторых, операции «`HAVE5`» и «`SUM`» не являются описанными выше операциями языка.

Задача E. Битовая Магия

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

У Егора есть любимое число n. Он обожает решать различные задачи, связанные с этим числом. Сегодня он подумал, что хочет найти максимальное целое число $m \leqslant n$, такое что двоичная запись числа m оканчивается **ровно** на k нулей.

Формат входных данных

В первой строке записано целое число n ($1 \leqslant n \leqslant 10^{18}$) — любимое число Егора.

Во второй строке записано целое число k ($0 \leqslant k \leqslant 60$) — необходимое количество нулей в конце двоичной записи числа m.

Формат выходных данных

Если искомого числа m не существует, выведите `-1`.

В противном случае выведите число m.

Примеры

стандартный ввод	стандартный вывод
10 3	8
5 4	-1
32 3	24

Замечание

В первом примеер число 8 — единственное число, не превосходящее 10, которое оканчивается на три нуля в двоичной записи ($8_{10} = 1000_2$).

Во втором примере нет ни одного подходящего числа, оканчивающегося на четыре нуля в двоичной записи.

В третьем примере число 32 оканчивается на пять нулей в двоичной записи. Так как нужно найти число, оканчивающееся **ровно** на три нуля, максимальное подходящее число равно 24.

Задача F. Прочтите Условие Задачи

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Однажды одна из команд довольно неклассического университета участвовала в сборах по программированию. На одном из контестов была предложена задача с довольно большим условием, которое принялись одновременно читать двое из трех участников команды — гроссмейстер-геометр Захар и специалист в области ASCII-графики и парсеров Майк. Через некоторое время ребята прочитали условие, обсудили его и выяснили, что задача является довольно простой, и через пятнадцать минут решение уже было готово и отправлено в тестирующую систему. Однако, по какой-то причине, команду ожидала неудача — «Неправильный ответ на тесте 4».

Майк и Захар обсудили решение, после чего их недоумение лишь возросло, так как они были уверены в правильности написанного кода. После этого Захар решил еще раз прочитать условие задачи, и оказалось, что написанный код действительно решал задачу... Только вот совсем другую, из-за того, что условие было прочитано невнимательно.

После этого герои данной истории вместе с третьим участником команды Никитой, мастером алгоритмов на графах, придумали решение правильной задачи и начали его реализовывать, ну а вам предстоит решить задачу в таком виде, в каком она была прочитана в первый раз. Формальная постановка данной задачи приведена ниже.

Дан неориентированный граф, состоящий из n вершин и m ребер, соединяющих некоторые пары вершин.

Назовем последовательность вершин $v_1, v_2, \ldots, v_k$ *простым путем*, если для любых $i \neq j$ верно, что $v_i \neq v_j$, а также для любого $i = 1 \ldots k-1$ в графе есть ребро, соединяющее вершины v_i и v_{i+1}.

Длиной простого пути будем считать количество вершин в нем.

Два простых пути $v_1, v_2, \ldots, v_k$ и $u_1, u_2, \ldots, u_l$ считаются *различными*, если $k \neq l$, либо $k = l$ и существует такое i, что $v_i \neq u_i$.

Назовем три различных простых пути 3-*связанными*, если выполнены следующие условия:

1. Первые вершины путей совпадают;
2. Последние вершины путей совпадают;
3. Если рассмотреть множество всех вершин данных путей, за исключением первой и последней, то найдется вершина, которая встречается более, чем в одном из трех путей.

Обратите внимание, что 3-связанные пути не обязаны иметь одинаковую длину.

Про данный граф известно, что в нем невозможно выбрать три различных простых 3-связанных пути.

Требуется для каждого $l = 1 \ldots n$ вычислить количество различных простых путей длины l в данном графе.

Формат входных данных

В первой строке через пробел записаны два целых числа n и m ($1 \leqslant n \leqslant 4\,000$, $0 \leqslant m \leqslant 10^5$) — количество вершин и ребер в графе, соответственно.

В каждой из следующих m строк через пробел записаны два числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant n, u_i \neq v_i$), означающие, что в графе вершины u_i и v_i соединены ребром.

Гарантируется, что каждая пара вершин соединена не более, чем одним ребром.

Формат выходных данных

Выведите через пробел n чисел: $c_1, c_2, \ldots, c_n$, где c_i — остаток от деления количества простых путей длины i в данном графе на число $998\,244\,353$.

Примеры

стандартный ввод	стандартный вывод
3 2 1 2 2 3	3 4 2
3 3 1 3 2 3 1 2	3 6 6
4 2 1 2 3 4	4 4 0 0

Задача G. За Орду!

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Сегодня великий вождь Саурфанг поведет орков в бой, и орки планируют сражаться так яростно, как никогда прежде. Любой великий вождь знает, что для успешной атаки нужно тщательное планирование.

В бою будут участвовать n орков, которых необходимо разделить на отряды. Саурфанг считает, что важны не количество или численность отрядов, а высокий боевой дух. Для каждого орка известно, что сам он пойдет в атаку с боевым духом a_i. Однако если орка воодушевит его командир, то он уже будет сражаться более яростно с боевым духом b_i. Но здесь возникает небольшая сложность...

Система подчинения орков весьма сложна. У каждого орка может быть несколько командиров, а у каждого командира может быть много подчиненных. Известно лишь, что все орки разного возраста, и никогда младший орк не может командовать старшим.

Саурфанг решил, что каждый отряд будет устроен следующим образом: вождь назначит командира отряда, который пойдет впереди. За ним будет идти его подчиненный, следующим будет подчиненный второго орка, и так далее. При этом воодушевлены будут все орки, кроме командира отряда. Возможно, что отряд будет состоять лишь из одного невоодушевленного орка.

Осталось лишь разделить орков на отряды, так, чтобы каждый орк был ровно в одном отряде, а суммарный боевой дух орков был максимальным. Но поскольку Саурфанг — орк, а орки не любят решать задачи, он пойдет точить свой топор, а сформировать отряды предстоит вам.

Формат входных данных

В первой строке задано число орков n ($1 \leqslant n \leqslant 10^5$).

Далее идет n строк. В i-й строке задано число k_i — количество орков, подчиняющихся i-му, а затем k_i чисел — номера этих орков. Орки пронумерованы в порядке старшинства, поэтому каждое число в i-й строке строго больше i. Сумма k_i не превосходит $2 \cdot 10^5$.

В следующей строке содержатся n целых чисел a_i ($1 \leqslant a_i \leqslant 10^9$).

В последней строке содержатся n целых чисел b_i ($a_i \leqslant b_i \leqslant 10^9$).

Формат выходных данных

В первой строке выведите два числа — суммарный боевой дух орков при оптимальном разбиении на отряды, и количество отрядов s.

В следующих s строках выведите описания отрядов. В начале описания выведите количество орков в отряде t_j, а затем t_j чисел — номера орков по порядку, начиная с командира отряда.

Если оптимальных разбиений существует несколько, выберите любое.

Пример

стандартный ввод	стандартный вывод
4 2 2 3 1 4 1 4 0 1 1 1 1 1 2 3 4	9 2 2 1 3 2 2 4

Задача H. Строим Параллелограммы — 2

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Представьте, что у вас есть прямоугольная сетка размера $N \times M$, разделенная на квадраты 1×1. От вас требуется посчитать, сколькими способами можно выбрать на данной сетке четыре узла таким образом, чтобы они образовывали невырожденный параллелограмм.

Напомним, что параллелограмм — это выпуклый четырехугольник, у которого противолежащие стороны параллельны друг другу. Параллелограмм называется невырожденным, если его площадь не равна нулю.

Формат входных данных

В единственной строке через пробел записаны два целых числа N и M ($1 \leqslant N, M \leqslant 10^6$, $1 \leqslant N \times M \leqslant 10^6$).

Формат выходных данных

Выведите одно целое число — количество параллелограммов.

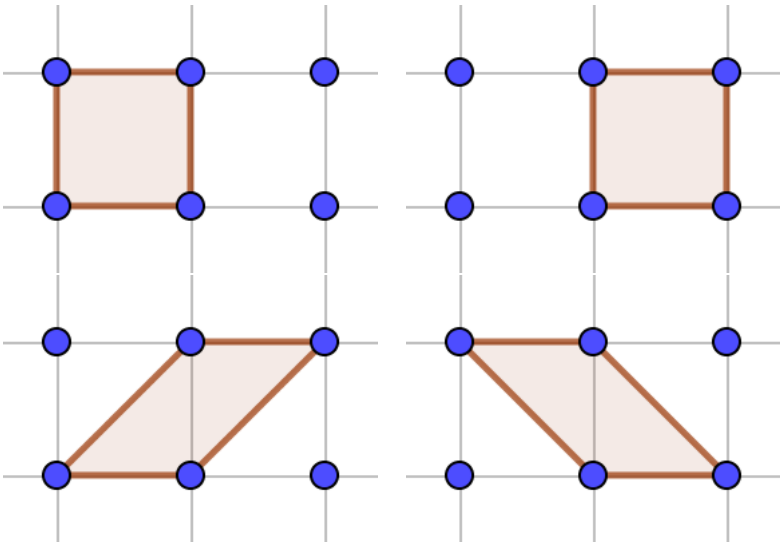
Примеры

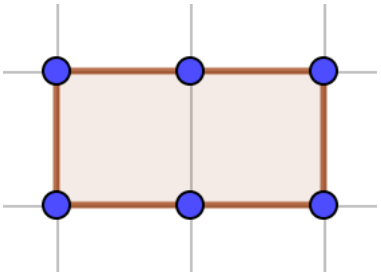
стандартный ввод	стандартный вывод
1 1	1
1 2	5
2 3	60
5 10	14267

Замечание

В первом примере сетка имеет размеры 1×1, у нее есть всего четыре узла, которые являются углами сетки. Соответственно, можно выбрать лишь один параллелограмм. В данном случае он также является квадратом 1×1.

Всевозможные варианты построения параллелограмма во втором примере изображены на рисунках ниже:





Здесь синим обозначены узлы сетки 1×2.

Задача I. Улучшение Навыков

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вы играете в необычную компьютерную игру, в которой основной целью является сражение с врагами и, конечно же, улучшение навыков вашего персонажа. Для улучшения навыков используются очки опыта. Недавно, одолев целую армию врагов, вы получили заветные N очков опыта, и теперь вы решили произвести улучшения. Но не все так просто! Для того, чтобы улучшить навыки персонажа, нужно выполнить несколько требований.

1. Вы можете улучшить уровень навыков вашего персонажа несколько раз. За каждое улучшение вы обязаны отдать некоторое ненулевое количество очков опыта. Сколько именно очков отдавать за каждое улучшение — решать вам.

2. Вы обязаны потратить все накопившиеся N очков опыта.

3. Представим, что вы улучшили навыки персонажа S раз, отдав за каждое улучшение $a_1, a_2, \ldots, a_S$ очков опыта, соответственно. Если найдутся такие l и r, что $1 \leqslant l \leqslant r \leqslant S$ и $a_l + a_{l+1} + \ldots + a_r = K$, то ваш персонаж моментально погибнет — это необычное свойство связано с черной магией, не пытайтесь понять, как это работает.

4. Конечно же, вы хотите улучшить навыки персонажа как можно больше раз.

К вам приближаются новые враги, поэтому нужно действовать быстро! Придумайте способ улучшить навыки вашего персонажа как можно больше раз, выполнив описанные выше требования. Конечно, в процессе улучшения навыков персонаж не должен погибнуть.

Формат входных данных

В единственной строке через пробел записаны два целых числа N и K ($1 \leqslant N \leqslant 2 \cdot 10^5$, $1 \leqslant K \leqslant 10^9$).

Формат выходных данных

В первой строке выведите одно число S — количество улучшений навыков вашего персонажа.

Во второй строке выведите последовательность из S натуральных чисел $a_1, a_2, \ldots, a_S$, где a_i — количество очков опыта, которые необходимо отдать за i-е улучшение.

Если невозможно улучшить навыки персонажа, соблюдая описанные правила, в единственной строке выведите число 0.

Примеры

стандартный ввод	стандартный вывод
1 2	1 1
5 3	2 4 1
4 4	0

Задача J. Строковый Ад

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В данной задаче вам предстоит окунуться в увлекательный мир строк!

Пусть имеется алфавит, состоящий из строчных латинских букв. Назовем *номером* буквы ее порядковый номер в алфавите и будем обозначать это как $num(a)$. Таким образом, $num(\texttt{a}) = 1$, $num(\texttt{b}) = 2$, а $num(\texttt{z}) = 26$.

Пусть у нас имеется две строки s и t одинаковой длины n: $|s| = |t| = n$. Будем обозначать i-й символ строки s как s_i.

Введем функцию $f(s, t)$, которая вычисляется по следующей формуле:

$$f(s,t) = \sum_{i=1}^{n} |num(s_i) - num(t_i)|$$

Дана строка s длины n. Обозначим подстроку строки s, начинающуюся с i-го символа строки и заканчивающуюся j-м символом строки, как $s[i \dots j]$.

От вас требуется посчитать следующую величину: зафиксируем некоторую длину подстроки $k = 1 \dots n$. Рассмотрим все пары подстрок строки s, имеющих длину k. Нужно вычислить сумму значений функции f от каждой пары подстрок.

Формально, нужно вычислить следующую сумму:

$$\sum_{k=1}^{n} \sum_{1 \leqslant i,j \leqslant n-k+1} f(s[i \dots i+k-1], s[j \dots j+k-1])$$

Формат входных данных

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

Во второй строке входных данных записана строка s длины n, состоящая из строчных латинских букв.

Формат выходных данных

Выведите единственное число — остаток от деления ответа на число 998 244 353.

Примеры

стандартный ввод	стандартный вывод
3 abb	6
4 bcaa	36
6 abcdef	252

Замечание

Рассмотрим ответ для первого примера.

В данной строке имеется три подстроки длины 1: `a`, `b` и `b`. В качестве примера, $f($`a`, `a`$) = 0$, $f($`a`, `b`$) = 1$. Тогда сумма значений функции для всех пар строк длины 1 будет равна: $0 + 1 + 1 + 1 + 0 + 0 + 1 + 0 + 0 = 4$.

Теперь рассмотрим все подстроки длины 2: `ab` и `bb`. Сумма значений равна $0 + 1 + 0 + 1 = 2$.

Теперь рассмотрим подстроки длины 3: `abb`. Сумма равна 0.

Тогда ответ равен $4 + 2 + 0 = 6$.

Задача K. Подсчет Графов

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Маша увлекается математикой и программированием, ее уже давно не удивить такими словами, как «Дерево отрезков», «Двойное тестирование», «Пересечение полуплоскостей», или, например, словом «Граф».

Напомним, что *графом* называется множество пронумерованных вершин, некоторые из которых соединены ребрами. Будем считать, что никакая вершина не должна быть соединена сама с собой, а также что любые две вершины должны быть соединены друг с другом не более, чем одним ребром.

Назовем *степенью вершины* графа количество ребер, которые соединяют данную вершину с какими-то другими вершинами.

И вот как-то раз Маше стало скучно решать задачи с двойным тестированием, пересекать полуплоскости, строить деревья отрезков и заниматься такими банальными вещами — ведь это так скучно! Вместо этого она придумала себе следующую задачу.

Обозначим за $C_k(n)$ количество всевозможных графов, состоящих из n вершин, в которых нет ни одной вершины со степенью k. Маше стало интересно, каких графов больше: графов без вершин со степенью 0, или же графов без вершин со степенью $n-1$? Иными словами, Маша захотела вычислить разность $C_0(n) - C_{n-1}(n)$ для некоторого выбранного n. Попробуйте сделать это и вы!

Формат входных данных

В единственной строке записано число n ($1 \leqslant n \leqslant 5\,000$) — количество вершин в исследуемых графах.

Формат выходных данных

Выведите одно число — ответ на задачу Маши.

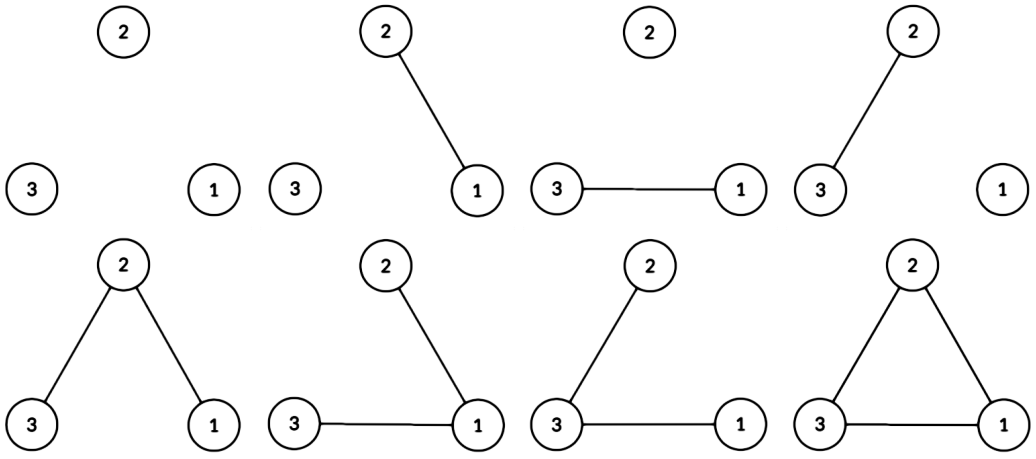
Нетрудно понять, что величина $C_k(n)$ может быть довольно большой, поэтому выведите остаток от деления числа $C_0(n) - C_{n-1}(n)$ на $998\,244\,353$.

Пример

стандартный ввод	стандартный вывод
3	0

Замечание

Ниже приведены всевозможные графы на трех вершинах.



Задача L. Конструирование Резисторов

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

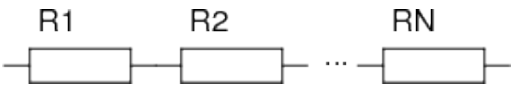
Вы любите физику? Впрочем, это неважно, мы ее обожаем! В этой задаче вам предстоит столкнуться с резисторами, а также с различными способами их соединения.

Напомним, что основной характеристикой резистора является его *сопротивление*. Резисторы можно объединять друг с другом в *цепь*, после чего общее сопротивление цепи вычисляется по общеизвестным формулам. В данной задаче мы будем рассматривать два типа соединения резисторов: последовательное и параллельное.

Пусть у нас имеются резисторы с сопротивлениями $R_1, R_2, \ldots, R_N$. Тогда при их последовательном соединении общее сопротивление полученной цепи будет равно:

$$R = R_1 + R_2 + \ldots + R_N$$

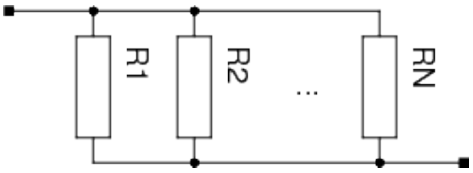
Такая цепь изображена на рисунке ниже:



При параллельном соединении резисторов общее сопротивление цепи вычисляется по формуле:

$$R = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_N}}$$

Такая цепь изображена на рисунке ниже:



Представим теперь, что у нас имеется специализированный компьютер для расчетов, связанных с электрическими цепями. У этого компьютера есть 128 регистров, в каждом из которых можно сохранить ровно одну электрическую цепь, состоящую только из резисторов. Регистры пронумерованы числами от 0 до 127. Изначально в регистре с номером 0 записана элементарная цепь: один резистор с сопротивлением $R_0 = 1$. Все остальные регистры изначально пусты.

Перед началом работы вы можете заполнить некоторое количество регистров с номерами от 1 до 127 *корректными электрическими цепями*. Электрическая цепь является корректной для записи в регистр с номером i, если:

- Для конструирования данной цепи используются только операции последовательного и параллельного соединения некоторых других цепей;
- Все используемые при соединении цепи были получены в регистрах с номерами, меньшими, чем i (включая регистр 0).

Для удобства обозначим общее сопротивление цепи, полученной в i-м регистре, как R_i.

После того, как вы заполнили необходимое вам количество регистров, к вам будут поступать запросы. Входным параметром каждого запроса является значение сопротивления $R = \frac{x}{y}$. В качестве ответа на запрос вы должны сконструировать электрическую цепь, общее сопротивление

которой в точности равно R. Для конструирования цепи вы можете использовать последовательные и параллельные соединения записанных в регистрах цепей. Более того, количество использованных цепей, сохраненных в регистрах, не должно быть слишком большим. Сможете ли вы справиться с поставленной задачей?

Протокол взаимодействия

Для начала вы должны вывести целое число t — количество регистров, которые вы собираетесь использовать ($1 \leqslant t \leqslant 128$). Это означает, что вы сможете воспользоваться регистрами с номерами от 0 до $t - 1$.

После этого выведите t строк, в каждой из которых выведите описание сконструированной цепи. Цепь, выведенная в $i + 1$-й строке, будет сохранена в регистр с номером i, после чего данную цепь можно будет использовать для построения новых цепей. Сопротивление данной цепи будет обозначено как R_i.

Описание каждой цепи должно быть задано в следующей грамматике (для лучшего понимания смотрите примеры):

⟨*circuit*⟩ ::= «R*i*» | ⟨*series*⟩ | ⟨*parallel*⟩
⟨*series*⟩ ::= «S(» ⟨*circuit*⟩ «,» ⟨*circuit*⟩ «)»
⟨*parallel*⟩ ::= «P(» ⟨*circuit*⟩ «,» ⟨*circuit*⟩ «)»

При этом при построении цепи, сохраняемой в регистр с номером i, разрешается использовать R_j только в случае $j < i$.

После этого вы должны считать число q ($1 \leqslant q \leqslant 1\,000$) — количество запросов.

Для ответа на каждый запрос сначала считайте два целых числа: x и y ($1 \leqslant x \leqslant 10^{18}$, $1 \leqslant y \leqslant 10^{18}$). После этого выведите описание построенной цепи в описанной выше грамматике. Общее сопротивление полученной цепи должно быть равно $\frac{x}{y}$. При описании цепи разрешается использовать R_i для всех $0 \leqslant i < t$.

Для всех построенных цепей, включая описание регистров, должны быть выполнены следующие ограничения. **Описание цепи должно быть записано на одной строке.** При описании цепи разрешается использовать сколько угодно пробелов, однако нельзя ставить пробелы между символом R и номером регистра. Количество использованных R_i в описании цепи не должно превышать 90. Длина полученного выражения не должна превышать 10 000 символов.

Не забудьте сделать операцию `flush` после каждого ответа на запрос, а также после вывода информации о регистрах в начале общения с интерактором.

В случае несоблюдения протокола взаимодействия вы можете получить любой вердикт!

Пример

стандартный ввод	стандартный вывод
3 2 5 6 3 17 4	4 S(R0, R0) P(R0, R0, R0) S(R1, R1) P(R1, R1, R1, R1, R1) R1 S(R3, P(R0, R0, P(R0, R0)))

Замечание

Рассмотрим пример. В начале мы сообщаем, что будем использовать только четыре регистра: R_0, R_1, R_2 и R_3. Изначально $R_0 = 1$. Далее мы задаем значения еще трех регистров.

В качестве R_1 берется последовательное соединение двух цепей R_0. Получаем общее сопротивление $R_1 = R_0 + R_0 = 1 + 1 = 2$.

В качестве R_2 берется параллельное соединение трех цепей R_0. Получаем общее сопротивление $R_2 = \dfrac{1}{\dfrac{1}{R_0}+\dfrac{1}{R_0}+\dfrac{1}{R_0}} = \dfrac{1}{\dfrac{1}{1}+\dfrac{1}{1}+\dfrac{1}{1}} = \dfrac{1}{3}$.

В качестве R_3 берется последовательное соединение двух цепей R_1. Получаем сопротивление $R_3 = R_1 + R_1 = 2 + 2 = 4$.

Далее интерактор сообщает нам три запроса.

В первом запросе нужно построить сопротивление $\dfrac{2}{5}$. Для этого мы решили соединить параллельно пять схем R_1. Получается, что сопротивление равно: $\dfrac{1}{\dfrac{1}{R_1}+\dfrac{1}{R_1}+\dfrac{1}{R_1}+\dfrac{1}{R_1}+\dfrac{1}{R_1}} = \dfrac{1}{\dfrac{1}{2}+\dfrac{1}{2}+\dfrac{1}{2}+\dfrac{1}{2}+\dfrac{1}{2}} = \dfrac{1}{\dfrac{5}{2}} = \dfrac{2}{5}$.

Во втором запросе требуется получить сопротивление $\dfrac{6}{3} = 2$. Для этого можно просто воспользоваться схемой R_1.

В третьем запросе необходимо построить сопротивление $\dfrac{17}{4}$. Для этого соединим последовательно схему R_3 с четырьмя параллельно соединенными схемами R_0. Получаем сопротивление $R_3 + \dfrac{1}{\dfrac{1}{R_0}+\dfrac{1}{R_0}+\dfrac{1}{R_0}+\dfrac{1}{R_0}} = 4 + \dfrac{1}{\dfrac{1}{1}+\dfrac{1}{1}+\dfrac{1}{1}+\dfrac{1}{1}} = 4 + \dfrac{1}{4} = \dfrac{17}{4}$.