Задача A. Транспорт Мюнхена

Имя входного файла:	`munich.in`
Имя выходного файла:	`munich.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В общественном транспорте Мюнхена используется несколько типов билетов. Дневной билет для одного взрослого действителен на неограниченное число поездок с момента его первого использования и до полуночи. Обозначим его стоимость как p_1. Аналогичный билет на ребенка от 5 до 14 лет стоит p_2 (но очевидно, что ребенок имеет право ездить и по билету взрослого).

По групповому билету могут одновременно ехать до 5 взрослых, а также каждый взрослый в этой группе может быть заменен на одного или даже двоих детей. Так, допустимыми являются группы из двух взрослых и 6 детей или 10 детей. Такой билет стоит p_3. Существуют также аналогичные виды билетов на 3 дня стоимостью q_1, q_2 и q_3 соответственно.

Группа из M взрослых и N детей указанного выше возраста прибыла в Мюнхен на K ($1 \le K \le 3$) дней. Они всегда будут перемещаться по городу вместе. Какие билеты им следует купить, чтобы потратить минимальную сумму денег.

Формат входного файла

Первая строка входных данных содержит числа M, N, K. Вторая строка –Ц числа p_1, p_2 и p_3. Третья –Ц числа q_1, q_2 и q_3. Все числа (кроме K) находятся в диапазоне от 1 до 1000.

Формат выходного файла

Выведите сумму, которая должна быть потрачена на билеты.

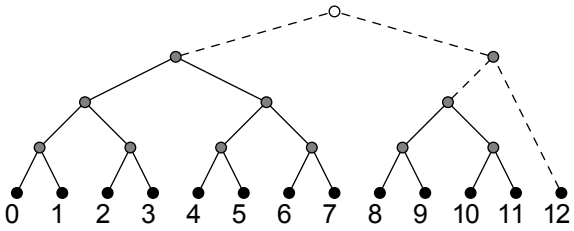
Пример

munich.in	munich.out
13 1 3 13 1 31 31 13 131	279

Задача B. Каноническое бинарное дерево

Имя входного файла:	`cbt.in`
Имя выходного файла:	`cbt.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Давайте опишем следующий алгоритм построения канонического бинарного дерева с N листьями (пример для $N = 13$ приведен на рисунке).



Элементы, пронумерованные слева направо от 0, разбиваются на пары. Парные элементы соединяются ребрами с общим предком. Предки первого уровня опять разбиваются на пары и т.д. Такие ребра на рисунке показаны сплошными линиями. В итоге, мы получим несколько полных поддеревьев (в том числе одно их них может состоять из одного элемента). На втором этапе полные бинарные поддеревья соединяются в одно дерево, начиная с правого поддерева: сначала соединяются два минимальных поддерева, потом опять два минимальных с учетом только что созданного и т.д. Ребра, созданные на втором этапе, показаны на рисунке пунктиром. Таким образом, каноническое бинарное дерево полностью описывается количеством листьев в дереве.

Как и в любом другом бинарном дереве, лист в каноническом дереве может быть однозначно идентифицирован или своим номером среди других листьев (см. числа 0 1 2 3 4 5 6 7 8 9 10 11 12 внизу рисунка) или путем от корня дерева к листу, описанному как последовательность слов *left* и *right*. Например, путь к листу номер 4 в показанном дереве будет *left*, *right*, *left*, *left*. Ваша задача написать программу, переводящую одно обозначение листа в другое (как в одну сторону, так и в другую, в зависимости от запроса).

Формат входного файла

Первая строка ввода состоит из двух целых: N ($1 < N < 2^{31}$) Ц число листьев в дереве, и Q ($1 \le Q \le 10000$) — число последующих запросов на перевод. Каждая из следующих Q строк описывает запрос. Каждый запрос начинается с буквы, соответствующей типу запроса. Затем через пробел идут параметры запроса.

Если тип запроса есть , то его единственный параметр это число L ($0 \le L \le N$), номер листа в нумерации слева направо, начиная с 0.

Если тип запроса , то параметры описывают путь от корня до листа как последовательность букв L или R.

Формат выходного файла

Выходные данные должны состоять в точности из Q строк, по одной на каждый запрос. Для запроса типа выходная строка должна содержать последовательность букв L и R, описывающих путь от корня к листу L. Для запроса типа B, строка должна содержать одно целое число — номер соответствующего пути листа.

Пример

cbt.in	cbt.out
13 2 A 4 B LRLL	LRLL 4

Задача C. Пример

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

Как вы уже наверное заметили, условие задач в соревнованию по программированию состоит из нескольких разделов. Наиболее важный раздел — это, конечно, примеры. Некоторые участники даже начинают читать условие задачи с примеров, а описание условия вообще читают последним. Это очень обидно авторам задач.

Поэтому на этот раз авторы решили описать примеры на том же языке, что и основное условие, соблюдая следующие правила.

- Целые положительные числа будут записываться в примере на английском языке (будем считать, что в примерах они меньше, чем 100). Слово `number` будет записан перед числом из примера, если оно не указывает на количество повторений. Например, `number zero`, `number sixteen`, или `number sixty one`.
- Последовательности символов (строки) будут записываться в двойных кавычках или в апострофах, например `"John's pen"`, or `'5" tall'`. Заметим, что апостроф может использоваться в строке, ограниченный кавычками и наоборот. Перед строкой пишется слово `string`, например `string 'Hello'`.
- Обозначение `space` соответствует одному пробелу.
- Число, строка или пробел могут начинаться с множителя, обозначающего количество их повторений. Множитель — это число больше единицы. Ему особые слова не предшествуют. Например, `four numbers five`, or `six strings 'A'`. Если множитель испльзуется по отношению к числу, то это означет соответствующее повторение данного числа через пробел. Так, уже приведенный пример означает `5 5 5 5`, а следующий — `AAAAAA`.
- Давайте назовем числа, строки или пробелы с соответствующим множителем фрагментом. Фрагменты могут быть организованы в последовательности с помощью словосочетания `followed by`. Например, `number five followed by number six`. Между числами, числом и строкой, строкой и числом ставится в точности один пробел, поэтому приведенный пример означает `5 6`.
- В результате пример к задаче описывается строчка за строчкой. Первая строка начинается со слов `The first line...`. Следующие строки описываются или `The next line...` или `The next \# lines...`, где `#` — это число большее единицы. Пустые строки в примере описывается словом `is empty` или `are empty`. Содержание других строк описывается после слова `contains` или `contain`. Первая буква в предложении всегда заглавна. Предложение заканчивается точкой (`.`). Точка не отделена пробелом от предшествующего слова, но отделена по крайней мере одним пробелом от следующего слова

Формат входного файла

Входные данные представляют собой описание примера. Слова отделяются друг от друга произвольным числом пробелов и переводов строки. После последней точки пробелов нет. Общей размер входного файла не превосходит 1 мегабайта.

Формат выходного файла

Вывод должен соответствовать содержимому описанного примера. Общий размер вывода не должен быть больше 1 мегабайта.

Пример

example.in	example.out
The first line contains four numbers twenty eight. The next line is empty. The next two lines contain six strings '-'. The next line contains number four followed by number seventy seven followed by string 'meat' followed by three strings "!".	28 28 28 28 ------ ------ 4 77 meat!!!

Задача D. Вычеркивание двух

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

Задана строка S, состоящая из маленьких букв латинского алфавита. Сколько различных строк можно получить при помощи вычеркивания ровно двух символов из S?

Формат входного файла

Входной файл содержит строку S, записанную в первой и единственной строке файла. Длина строки S от 2 до 100000 символов включительно. Строка S содержит только маленькие буквы латинского алфавита.

Формат выходного файла

Выходной файл должен содержать одно целое число, равное количеству различных строк, которые можно получить при помощи вычеркивания ровно двух символов из S.

Пример

cuttwo.in	cuttwo.out
abbccc	5

Задача E. K-ое разбиение

Имя входного файла:	`kth.in`
Имя выходного файла:	`kth.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Рассмотрим разбиения натурального числа N на положительные слагаемые, удовлетворяющие следующим условиям:

$$\sum_{i=1}^{M} p_i = N$$

$$|p_i - p_j| \leq 1$$

Для любых i, j. При этом разбиения, отличающиеся только порядком слагаемых, будем считать различными. Все разбиения могут быть отсортированы в лексикографическом порядке в смысле сравнения полученных строк, в которых слагаемые разбиения разделены пробелом (пробел при этом считается меньше любой цифры). Например, все упорядоченные разбиения числа 5 выглядят так:

1 1 1 1 1
1 1 1 2
1 1 2 1
1 2 1 1
1 2 2
2 1 1 1
2 1 2
2 2 1
2 3
3 2
5

По данному N найдите K-е в лексикографическом порядке разбиение числа N.

Формат входного файла

Первая строка входных данных содержит два натуральных числа N ($1 \leq N \leq 60$), T ($1 \leq T \leq 1000$). T — количество различных тестовых наборов во входном файле, соответствующих указанному значению N. Каждая следующая из строк описывает один тест и содержит значение K ($1 \leq K \leq 10^{18}$).

Формат выходного файла

Для каждого теста в отдельной строке выведите K-е в лексикографическом порядке разбиение на слагаемые числа N. Если общее количество разбиение меньше, чем K, то выдайте 1.

Пример

kth.in	kth.out
13 3 1 13 1313	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 -1

Задача F. Тупики

Имя входного файла:	`deadlocks.in`
Имя выходного файла:	`deadlocks.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

На вокзале есть K тупиков, куда прибывают электрички. Этот вокзал является их конечной станцией, поэтому электрички, прибыв, некоторое время стоят на вокзале, а потом отправляются в новый рейс (в ту сторону, откуда прибыли).

Дано расписание движения электричек, в котором для каждой электрички указано время ее прибытия, а также время отправления в следующий рейс. Электрички в расписании упорядочены по времени прибытия. Поскольку вокзал — конечная станция, то электричка может стоять на нем довольно долго, в частности, электричка, которая прибывает раньше другой, отправляться обратно может значительно позднее.

Тупики пронумерованы числами от 1 до K. Когда электричка прибывает, ее ставят в свободный тупик с минимальным номером. При этом если электричка из какого-то тупика отправилась в момент времени X, то электричку, которая прибывает в момент времени X, в этот тупик ставить нельзя, а электричку, прибывающую в момент $X + 1$ — можно.

Напишите программу, которая по данному расписанию для каждой электрички определит номер тупика, куда прибудет эта электричка.

Формат входного файла

Сначала вводятся число K — количество тупиков и число N — количество электропоездов ($1 \le K \le 100000, 1 \le N \le 100000$). Далее следуют N строк, в каждой из которых записано по 2 числа: время прибытия и время отправления электрички. Время задается натуральным числом, не превышающим 10^9. Никакие две электрички не прибывают в одно и то же время, но при этом несколько электричек могут отправляться в одно и то же время. Также возможно, что какая-нибудь электричка (или даже несколько) отправляются в момент прибытия какой-нибудь другой электрички. Время отправления каждой электрички строго больше времени ее прибытия.

Все электрички упорядочены по времени прибытия. Считается, что в нулевой момент времени все тупики на вокзале свободны.

Формат выходного файла

Выведите N чисел — по одному для каждой электрички: номер тупика, куда прибудет соответствующая электричка. Если тупиков не достаточно для того, чтобы организовать движение электричек согласно расписанию, выведите два числа: первое должно равняться 0 (нулю), а второе содержать номер первой из электричек, которая не сможет прибыть на вокзал.

Примеры

deadlocks.in	deadlocks.out
1 1 2 5	1
1 2 2 5 5 6	0 2
2 3 1 3 2 6 4 5	1 2 1