Задача A. Пульт дистанционного управления

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

Несколько лет назад Петя приобрел новый телевизор фирмы “Berlony”. Телевизор поддерживает 100 каналов, которые нумеруются от 0 до 99. На пульте дистанционного управления для этого телевизора 13 кнопок:

```
1  2  3 ↑
4  5  6 ↓
7  8  9
-  0
```

При нажатии кнопки “↑” номер текущего канала увеличивается на 1 (если текущий канал 99, то новый канал будет 0). При нажатии кнопки “↓” номер текущего канала уменьшается на 1 (если текущий канал 0, то новый канал будет 99). Для непосредственного переключения на канал от 0 до 9 необходимо нажать цифру с соответствующим номером. Для непосредственного переключения на канал от 10 до 99 необходимо нажать кнопку “-”, а затем две кнопки, соответствующие цифрам этого канала. После долгого использования пульт дистанционного управления от телевизора фирмы “Berlony” сломался, поэтому некоторые кнопки на пульте не работают. Ваша задача за наименьшее количество нажатий кнопок переключиться с канала X на канал Y.

Формат входного файла

В первых 4 строках записана информация о работоспособности каждой кнопки пульта. 0 означает, что кнопка не работает, а 1 означает, что кнопка работает. В первой строке описываются кнопки “1”, “2”, “3” и “↑” соответственно. Во второй строке описываются кнопки “4”, “5”, “6” и “↓” соответственно. В третей строке описываются кнопки “7”, “8” и “8” соответственно. В четвертой строке описываются кнопки “–” и “0” соответственно. в 5 строке записано 2 числа X и Y.

Формат выходного файла

В выходной файл выведите ответ на задачу — минимальное количество нажатий кнопок, чтобы переключиться с канала X на канал Y. Выведите -1, если невозможно переключиться на канал Y.

Пример

remote.in	remote.out
1 1 1 1 1 1 1 1 1 1 1 1 1 23 52	3
0 0 1 1 1 1 1 1 1 1 1 1 1 23 52	4

Задача B. Кручу число

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

Профессор Далл изобрел новое направление в теории чисел. Теория «крутильных генераторов». Рассмотрим положительное число p. Пусть оно состоит из b бит (старший бит обязательно единичный). Рассмотрим все возможные b циклических сдвигов двоичной записи числа p. Образуем из этих сдвигов множество $W(p)$. Заметим, что некоторые из этих сдвигов могут начинаться с 0. Будем говорить, что число p крутильно порождает множество $W(p)$. Например, $W(11) = \{7, 11, 13, 14\}$. Число p называется крутильным генератором числа n, если объединение множеств $W(1), W(2), \ldots, W(p)$ должно содержать $\{1, 2, \ldots, n\}$. Найдите наименьший из всех генераторов числа n.

Формат входного файла

В первой строке входного файла записано целое число n ($1 \le n \le 10^{18}$).

Формат выходного файла

Выведите наименьший крутильный генератор числа n.

Пример

twist.in	twist.out
6	5

Задача C. Шарики

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

Для предстоящего праздника нужно как можно быстрее надуть M шариков. Для этого пригласили N добровольцев и арендовали аппарат для надувания шариков. В любой момент времени аппарат не может надувать более одного шарика. Было решено, что добровольцы будут работать на аппарате по одной минуте. Доброволец под номером i за одну минуту может надуть A_i шариков, после чего он должен отдохнуть не менее B_i минут. Необходимо найти минимальное время, за которое они могут надуть M шариков.

Формат входного файла

В первой строке записаны целые числа M и N ($1 \le M \le 100, 1 \le N \le 10$). Далее идут N строк по два целых числа A_i и B_i ($0 \le A_i \le 10, 1 \le B_i \le 4$).

Формат выходного файла

В первую строку выведите T - минимальное время, за которое добровольцы могут надуть M шариков (в минутах).

Пример

balls.in	balls.out
10 3 4 4 3 4 2 3	5

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. Дельта, Каппа, Лямбда

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

Рассмотрим связный неориентированный граф G.

Степень вершины v, обозначаемая как $degv$ это количество ребер, концом которых является эта вершина. Минимальная степень вершины в графе G обозначается как $\delta(G)$.

Вершинной связностью графа G, обозначаемой как $\kappa(G)$, называется минимальное число вершин, которые необходимо удалить из графа, чтобы он содержал как минимум две компоненты связности. Для полного графа K_n положим $\kappa(K_n) = n - 1$.

Реберной связностью графа G, обозначаемой как $\lambda(G)$, называется минимальное количество ребер, которые необходимо удалить из графа, чтобы он содержал как минимум две компоненты связности. Для графа с одной вершиной без ребер K_1 положим $\lambda(K_1) = 0$. По заданным δ, κ и λ постройте граф, для которого $\delta(G) = \delta$, $\kappa(G) = \kappa$ и $\lambda(G) = \lambda$, либо определите, что такого графа не существует.

Формат входного файла

Входной файл содержит три целых числа: δ, κ и λ ($1 \le \delta, \kappa, \lambda \le 10$).

Формат выходного файла

Первая строка выходного файла должна содержать два целых числа: n и m количество вершин и ребер в построенном графе, соответственно. Следующие m строк должны описывать ребра графа. Граф не должен содержать петель и кратных ребер. В графе должно быть не более 50 вершин.

Если искомого графа не существует, выведите на первой строке выходного файла два нуля.

Примеры

dkl.in	dkl.out
2 2 2	3 3 1 2 2 3 1 3
1 1 2	0 0

Задача F. Пастух

Имя входного файла:	`shepherd.in`
Имя выходного файла:	`shepherd.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	64 мегабайта

Антон — пастух. У него есть n овец, которые спокойно жуют травку на пастбище. А Антон сидит себе на холме, играет на дудочке, да наблюдает за своими овцами.

Но в соседнем лесу водятся злые волки! Овцы могут быть в опасности! Поэтому Антон очень внимательно следит, чтобы его овцы были в полной безопасности. Он хотел бы сидеть так, чтобы угол, под которым он видит овец, был как можно меньше. Разумеется, угол, под которым он смотрит на овец не должен превышать 180 градусов. Но, к сожалению, зрение у Антона довольно среднее, поэтому он плохо видит овцу, если она находится на расстоянии более D от него. Поскольку он должен хорошо видеть своих овец, никакая овца не должна быть дальше чем D от Антона.

Чтобы овцы не нервничали от присутствия человека, Антон не должен быть ближе d ни к какой из своих овец.

По информации о расположении овец, величинам D и d, определите, где следует находится Антону, чтобы угол, под которым он видел овец был как можно меньше (и не превышал 180 градусов).

Формат входного файла

Первая строка входного файла содержит три целых числа: n, d и D ($3 \le n \le 10, 1 \le d \le D \le 10^3$).

Следующие n строк содержат по два целых числа — координаты овец. Координаты не превышают 10^3 по абсолютной величине. Никакие три овцы не находятся на одной прямой.

Формат выходного файла

Выведите два вещественных числа — координаты точки, где следует сидеть Антону, чтобы минимизировать угол, под которым он видит овец. Если решения не существует, выведите слово `impossible`. Проверяющая программа будет использовать точность 10^{-5} для сравнения вещественных чисел. Выводите как можно больше знаков после десятичной точки.

Пример

shepherd.in	shepherd.out
4 1 5 0 0 1 0 0 1 1 1	4.9749371855331 0.5

Задача G. 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

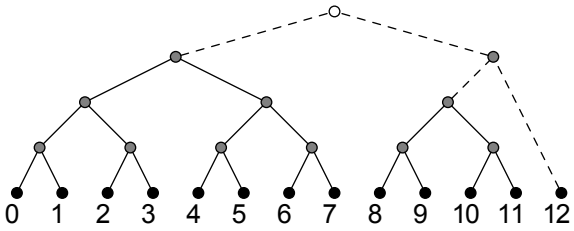
`munich.in`	`munich.out`

`13 1 3`
`13 1 31`
`31 13 131`
	`279`

Задача H. 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 –Ц номер соответствующего пути листа.

Пример

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

Задача I. 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 J. 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