Задача A. Мероприятие

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

Мише поручили решить следующую задачу: имеется дата начала и конца некоторого мероприятия; требуется определить его длительность. Он написал программу и попросил Машу её проверить.

Через некоторое время пришла Маша и расстроила Мишу: «Твоя программа работает неправильно. По-моему, ты забыл, что года бывают високосными.»

У Миши очень мало свободного времени и он не успевает исправить свою программу. Помогите ему.

Год является високосным, тогда и только тогда, когда выполнено одно из следующих условий:

- год делится на 4, но не делится на 100;
- год делится на 400.

Например, года 400, 404, 496, 504, 2000, 2004 являются високосными, а года 100, 200, 300, 503, 1000, 2001, 2005 - нет.

Формат входного файла

Первая строка входного файла содержит дату начала мероприятия. Вторая строка входного файла содержит дату конца мероприятия. Гарантируется, что первая дата меньше второй.

Даты заданы в формате `DD.MM.YYYY`.

Формат выходного файла

Выведите единственное число - длительность мероприятия (в днях).

Примеры

arrange.in	**arrange.out**
`01.09.2005` `06.10.2005`	`36`
`01.09.0005` `01.09.0006`	`366`
`01.02.2004` `01.03.2004`	`30`

Задача B. Коробки

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

На столе лежат коробка размера $A_1 \times B_1 \times C_1$ и коробка размера $A_2 \times B_2 \times C_2$. Выясните можно ли одну из этих коробок положить в другую, если разрешены повороты коробок вокруг любого ребра на угол 90 градусов.

Формат входного файла

Первая строка входного файла содержит три целых числа A_1, B_1 и C_1. Вторая строка входного файла содержит три целых числа A_2, B_2 и C_2.

Все числа положительны и не превосходят 1000.

Формат выходного файла

Если коробки одинаковы, выведите “`Boxes are equal`”. Если первая коробка может быть положена во вторую, выведите “`The first box is smaller than the second one`”. Если вторая коробка может быть положена в первую, выведите “`The first box is larger than the second one`”. Иначе, выведите “`Boxes are incomparable`”.

Примеры

boxes.in	**boxes.out**
`1 2 3` `3 2 1`	`Boxes are equal`
`2 2 3` `3 2 1`	`The first box is larger than the second one`
`2 2 3` `3 2 3`	`The first box is smaller than the second one`
`3 4 5` `2 4 6`	`Boxes are incomparable`

Задача C. Скобки

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

Коля написал длинное арифметическое выражение и хочет проверить, не ошибся ли он. Но он не знает, как это сделать. Через некоторое время он решил упростить себе задачу: для начала проверить, является скобочная последовательность правильной. Помогите Коле.

Имеется последовательность открывающих и закрывающих скобок. Требуется выяснить, является ли она правильной.

Правильными называются последовательности скобок, удовлетворяющие следующим условиям:

- Пустая последовательность — правильная.
- Если A и B — правильные скобочные последовательности, то (A) и AB — правильные скобочные последовательности.
- Последовательности, которые нельзя получить применением конечного числа указанных выше правил, не являются правильными.

Формат входного файла

Первая строка входного файла содержит последовательность скобок.

Длина строки не прышает 100000 символов.

Формат выходного файла

Если последовательность является правильной, выведите «CORRECT». Иначе, выведите «INCORRECT».

Примеры

brackets.in	brackets.out
`()(((()())()))`	CORRECT
`()()())(()`	INCORRECT

Задача D. Комментарии

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

Имеется программа, написанная на языке Delphi. Выведите её без комментариев и пустых строк (строка считается пустой, если она не содержит ни одного символа).

Всем известно, что программа на языке Delphi может содержить три вида комментариев:

- комментарий начинается с `{` и продолжается до ближайшей `}`;
- комментарий начинается с `(*` и продолжается до ближайших `*)`;
- комментарий начинается с `//` и продолжается до конца строки;

В этой задаче директивы компилятору будем считать обычными комментариями. Не забудьте, что комментарий не может начинаться внутри строки (строки заключены в одинарные кавычки).

Формат входного файла

Входной файл содержит программу с комментариями.

Размер входного файла не превосходит 10Кб.

Формат выходного файла

Выведите программу без комментариев и пустых строк.

Примеры

comments.in

```
//First type of comments
{Second type of comments}
(*Third type of comments*)

begin
writeln('{Normal string}');

( * Word1 * )

(* Word2// *) Word3

{ Word3 }

(* Word//4 *)
end.
```

comments.out

```
begin
writeln('{Normal string}');
( * Word1 * )
Word3
end.
```

Задача E. Кубик

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

Все мы в детстве играли в кубики. Были у нас и кубики с цифрами, и кубики с буквами. Были также и разноцветные кубики. Маленький Андрюша тоже любит играть в кубики. У него есть несколько наборов кубиков, причем все кубики из одного набора раскрашены одинаково, а кубики из разных наборов — по-разному.

На столе у Андрюши лежат два кубика. Помогите ему определить, принадлежат они одному набору или нет. Стол у Андрюши стеклянный, поэтому он видит цвета всех граней кубика.

Кубики принадлежат одному набору, если один из них можно комбинацией поворотов вокруг осей, проходящих через середины противоположных граней, перевести в другой.

Формат входного файла

Первая строка содержит описание первого кубика в формате: цвет передней грани, цвет задней грани, цвет верхней грани, цвет нижней грани, цвет левой грани, цвет правой грани.

Вторая строка — описание второго в таком же формате. Цвета граней кодируются числами. Все числа во входном файле — целые, положительные и не превосходят 100.

Формат выходного файла

Выведите в выходной файл «YES», если первый кубик и второй кубики принадлежат одному набору, и «NO» — в противном случае.

Примеры

cube.in	cube.out
1 2 3 4 5 6 1 2 3 4 5 6	YES
1 2 3 4 5 6 1 1 1 1 1 1	NO
1 2 3 4 5 6 5 6 3 4 2 1	YES

Задача F. Рисование

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

При обучении школьников младших классов программированию часто используется язык «Лого», позволяющий рисовать на экране картинки хвостом виртуальной черепашки.

Рассмотрим упрощенную версию этого языка, в которой разрешается подавать черепашке следующие команды: переместиться вперед на некоторое количество сантиметров, рисуя за собой линию, повернуть налево на 90 градусов и повернуть направо на 90 градусов.

Рассмотрим замкнутую фигуру, нарисованную таким образом. Требуется проверить, верно ли, что любой вертикальный и любой горизонтальный отрезок, соединяющий две точки фигуры, полностью содержится внутри этой фигуры.

Формат входного файла

Первая строка входного файла содержит число n — количество команд черепашки, использованных для рисования фигуры ($7 \le n \le 40\,000$). Следующие n строк содержат команды.

Команды задаются следующим образом:

- “`f` x” — переместиться вперед на x сантиметров (x — натуральное число);
- “`l`” — повернуть налево;
- “`r`” — повернуть направо.

Известно, что последовательность команд начинается с команды перемещения, не выполняется подряд две команды поворота или две команды перемещения, последняя команда также является командой перемещения.

Гарантируется, что ломаную можно поместить в некоторый прямоугольник, длины сторон которого не превосходят 10^9.

Формат выходного файла

В выходной файл выведите «`TRUE`», если нарисованная фигура удовлетворяет описанному выше свойству и «`FALSE`», если не удовлетворяет.

Примеры

drawing.in	**drawing.out**
11 f 1 r f 2 r f 2 r f 1 r f 1 l f 1	TRUE

Задача G. Спираль

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

В одной из предыдущих задач уже описывался язык «Лого» для рисования простейших фигур. Теперь вам предлагается написать на этом языке программу, изображающую контур спирали.

Спираль — это множество клеток, которое строится по следующим правилам. Рассмотрим прямоугольник, состоящий из $m \times n$ единичных клеток. Закрасим некоторые из его клеток в следующией последовательности.

Начнем красить левый столбец снизу вверх. Когда продвижение невозможно, то есть либо следующая клетка лежит вне прямоугольника, либо соприкасается с закрашенной ранее, направление движения поворачивается по часовой стрелке на 90 градусов. Если и после этого поворота дальнейшее движение невозможно, процесс заканчивается.

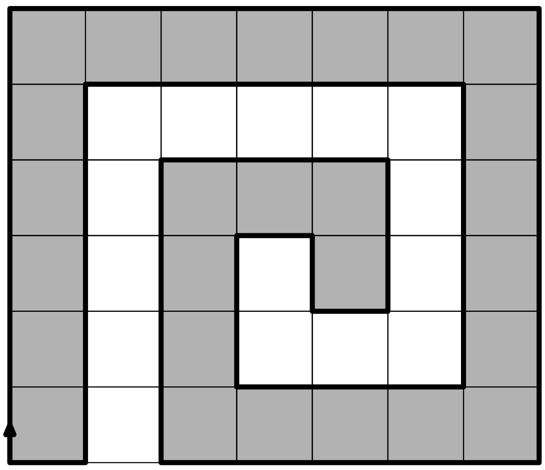
Множество закрашенных таким образом клеток и образует спираль.

Формат входного файла

Во входном файле записаны два целых числа m и n ($1 \le m, n \le 20\,000$).

Формат выходного файла

В выходной файл выведите последовательность команд, в результате выполнения которой будет нарисован контур ломаной. Напомним формат, в котором следует выводить результат. Первая строка должна содержать k — количество команд. Последующие k строк задают либо длину отрезка d, который надо провести, в формате `f` d, либо направление поворота — `l` для поворота налево и `r` для поворота направо. Команды рисования отрезка и поворота должны чередоваться, начинаясь и заканчиваясь командой рисования отрезка. Считайте, что начало первого отрезка совпадает с левым нижним углом прямоугольника. При этом контур должен быть выведен в направлении обхода по часовой стрелке, и никакая его часть ненулевой длины не должна быть нарисована дважды.



Примеры

spiral.in	**spiral.out**
6 7	31 f 6 r f 7 r f 6 r f 5 r f 4 r f 3 r f 2 r f 1 r f 1 l f 1 l f 2 l f 3 l f 4 l f 5 l f 5 r f 1

Задача H. Сундук с сокровищами

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

Путешественник нашел карту, указывающую расположение сундука с сокровищами, и захотел его найти. Карта оказалась настолько старой, что он не смог определить место, где были спрятаны сокровища. К счастью, на другой стороне пергамента был написан один из возможных маршрутов, идя по которому можно добраться до искомого сундука.

Маршрут начинается в точке (0, 0) и состоит из k инструкций. Для каждой инструкции указано направление движения dir_i (`NORTH`, `SOUTH`, `WEST`, `EAST`, `NORTHWEST`, `NORTHEAST`, `SOUTHWEST`, `SOUTHEAST`) и расстояние $dist_i$, которое необходимо пройти в этом направлении.

Напишите программу, которая по маршруту определяет местоположение сундука.

Считайте, что ось Ox направлена с запада на восток, а ось Oy — с юга на север.

Формат входного файла

Первая строка входного файла содержит единственное число k ($0 \le k \le 10000$) — количество инструкций в маршруте. Следующие k строк содержат по одной инструкции. Каждая инструкция представлена в следующем формате: направление движения dir_i, один пробел и вещественное число $dist_i$ ($0 \le dist_i \le 10000$) — расстояние, которое необходимо пройти в указанном направлении.

Формат выходного файла

На единственной строке выходного файла выведите два числа x, y с точностью 4 знака после запятой — координаты сундука.

Примеры

treasure.in	**treasure.out**
2 NORTHEAST 1.4142 WEST 0.3	0.6999904 0.9999904
3 NORTH 10 WEST 10 SOUTHEAST 10	-2.9289322 2.9289322