Задача A. Принтер

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

Завтра у Пети доклад на уроке истории. Для доклада Петя подготовил великолепный текст, а также взял несколько иллюстраций в библиотеке.

Но вот незадача: текст у Пети находится в электронном виде, а значит, его нужно распечатать. К несчастью, в его принтере кончаются чернила, и на доклад их может не хватить. В таком случае ему придется посылать папу в магазин за новым картриджем (впрочем, может оказаться, что не Петя пошлет папу в магазин, а наоборот). Чтобы не гонять папу (а, может быть, и не папу) зря, Петя решил подсчитать, хватит ли оставшихся чернил на его доклад.

Чтобы это сделать, Петя для каждой буквы оценил количество чернил, требуемое для того, чтобы ее напечатать. Более того, некоторые понятия в тексте Пети выделены с помощью заглавных букв, и это тоже надо учитывать.

Помогите Пете подсчитать суммарное количество чернил, необходимое для того, чтобы распечатать текст его доклада.

Формат входного файла

В первой строке входного файла содержится число N ($1 \le N \le 26$) используемых в тексте букв латинского алфавита.

В каждой из последующих строк содержится символ S_i и два целых числа A_i и B_i. S_i — это строчная буква латинского алфавита, A_i — количество чернил, требуемое для печати строчной буквы, B_i — количество чернил, требуемое для печати соответствующей заглавной буквы. Все S_i различны. Для чисел A_i и B_i верны неравенства $1 \le A_i, B_i \le 100$.

В следующей строке находится текст доклада Пети. Его длина не превосходит 100 символов. Текст состоит из строчных и заглавных букв латинского алфавита, причем в тексте присутствуют только те буквы, для которых во входом файле задано требуемое для их печати количество чернил.

Формат выходного файла

В выходной файл выведите суммарное количество чернил, требуемое для вывода текста на печать.

Примеры

printer.in	printer.out
5 p 5 6 t 7 8 y 9 10 a 1 2 e 3 4 Petya	26
3 a 2 8 b 4 6 c 5 7 abaCaba	23

Задача B. Игра Set

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

Игра Set — это карточная игра, развивающая навыки зрительного восприятия. Для нее необходима специальная колода из 81 карты. Все фигуры, изображенные на картах, различаются по четырем признакам:

- число — на карте может быть изображена одна (`one`), две (`two`) или три фигуры (`three`);
- цвет — фигуры могут быть красными (`red`), зелеными (`green`) или фиолетовыми (`purple`);
- заливка — может отсутствовать (`open`), быть сплошной (`solid`) или «в полосочку» (`striped`);
- тип — на карте могут изображены овалы (`oval`), ромбы (`diamond`) или тильды (`squiggle`).

Цель игры состоит в поиске в картах, которые выкладываются на стол так называемых *сетов*. Сет — это набор из трех карт, которые по каждому из четырех признаков либо все одинаковы, либо все различны.

Задан набор попарно различных карт. Необходимо найти число троек карт (не обязательно непересекающихся) из этого набора, которые образуют сеты.

Формат входного файла

Первая строка входного файла содержит число карт n ($3 \le n \le 81$). Каждая из последующих n строк описывает одну из карт. Описание каждой карты состоит из четырех слов на английском языке — первое из них обозначает число фигур на карте, второе — цвет, третье — заливку, а четвертое — тип. При этом если фигура на карте одна, то ее тип указан в единственном числе (`oval`, `diamond`, `squiggle`), а если не одна, то во множественном (`ovals`, `diamonds`, `squiggles`).

Формат выходного файла

В выходной файл выведите ответ на задачу.

Примеры

set.in	set.out
3 one red solid oval two green solid ovals three purple solid ovals	1
3 one red open oval two green open diamonds three purple striped squiggles	0

Задача C. Битоническая подпоследовательность

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

Последовательность чисел b_1, b_2, …, b_m называется *битонической*, если существует такое число j $(1 < j < m)$, что выполняются неравенства $b_1 < b_2 < \ldots < b_j > b_{j+1} > \ldots > b_m$. Заметим, что, в соответствии с этим определением, битоническая последовательность должна содержать хотя бы три элемента.

Пусть задана некоторая последовательность чисел a_1, a_2, …, a_n. Ее подпоследовательностью называется последовательность следующего вида:

$$a_{i_1}, a_{i_2}, \ldots a_{i_k}$$

При этом для чисел i_1, …, i_k должны выполняться неравенства $1 \le i_1 < i_2 < \ldots < i_k \le n$.

Ваша задача состоит в том, чтобы написать программу, которая найдет битоническую подпоследовательность заданной последовательности, для которой максимальна сумма цифр входящих в нее чисел. При этом предполагается, что числа записываются в десятичной системе счисления.

Формат входного файла

Первая строка входного файла содержит целое число n $(3 \le n \le 1000)$. Вторая строка входного файла содержит n целых чисел a_1, …, a_n — заданную последовательность. Для каждого из них верны неравенства $1 \le a_i \le 10^9$.

Формат выходного файла

В первой строке выходного файла выведите сумму цифр чисел, входящих в найденную битоническую подпоследовательность. Если у заданной последовательности нет битонических подпоследовательностей, выведите в выходной файл число `-1`.

Примеры

bitonic.in	bitonic.out
4 2 1 3 1	6
3 1 2 3	-1

В первом примере у заданной последовательности существуют две битонические подпоследовательности: `2 3 1` и `1 3 1`. У первой сумма цифр входящих в нее чисел больше, чем у второй.

Задача D. Симулятор железной дороги

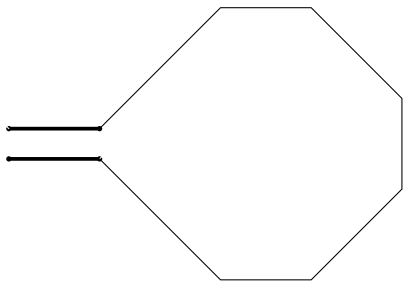
Имя входного файла:	`railroad.in`
Имя выходного файла:	`railroad.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

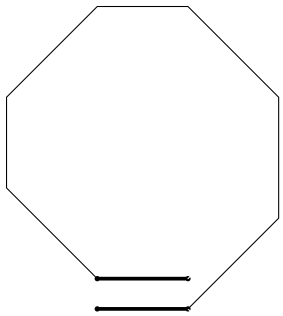
Петя играет на компьютере в увлекательную игру — симулятор железной дороги. Для начала он построил две станции и теперь хочет соединить их железной дорогой, по которой будет ездить поезд с n вагонами.

Игра происходит на бесконечной плоскости. Станции и железные дороги представляют собой отрезки с концами в точках с целочисленными координатами. Станция для поезда с n вагонами представляет собой отрезок длины n, параллельный одной из осей координат. Железная дорога, соединяющая две станции, должна быть ломаной, отрезки которой либо параллельны осям координат, либо проходят к ним под углом в 45°. Концы ломаной должны совпадать с концами отрезков-станций (любой конец одной станции должен быть соединен с любым концом другой). Железная дорога может пересекать себя или станции (Петя может построить в соответствующих местах мосты или туннели).

Чтобы поезд мог без задержек ездить по железной дороге, все углы ломаной должны быть равны 135° — прямые углы по 90° или острые по 45° не допускаются. Аналогично, не должно быть прямых или острых углов между станцией и железной дорогой. Поезд не должен пытаться одновременно выполнить два поворота, одновременно выезжать или заезжать на станцию и выполнять поворот или одновременно выезжать со станции и заезжать на станцию. Поэтому каждый отрезок железной дороги должен иметь длину хотя бы n.

Два приведенных рисунка показывают две станции, соединенные железной дорогой. Вариант на правом рисунке имеет более короткую железную дорогу.





Петя разместил станции и теперь хочет соединить их кратчайшей возможной железной дорогой. Помогите ему это сделать.

Формат входного файла

Первая строка входного файла содержит n ($1 \le n \le 10$). Вторая строка содержит четыре целых числа: x_1, y_1, x_2 and y_2 — координаты концов первой станции. Координаты целые и отрезок $(x_1, y_1) - (x_2, y_2)$ параллелен одной из координатных осей. Третья строка содержит описание второй станции в том же формате. Все координаты от 0 до 50. Станции не имеют общих точек.

Формат выходного файла

Первая строка выходного файла должна содержать l — количество отрезков ломаной, которая представляет железную дорогу. Следующие $l + 1$ строка должны содержать по два целых числа — концы и точки излома ломаной.

Примеры

railroad.in	railroad.out
3 0 0 3 0 0 1 3 1	7 3 0 6 3 6 7 3 10 0 10 -3 7 -3 4 0 1