Задача A. Наидлиннейшая 2-подстрока

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дана строка s, которая состоит из n строчных букв латинского алфавита. Ее подстрока называется 2-подстрокой, если она содержит не более двух различных букв. Например, в строке «`cabacaa`» подстроки «`aba`», «`c`» и «`acaa`» — 2-подстроки, а подстроки «`cab`», «`baca`» — нет.

Ваша задача вывести самую длинную 2-подстроку заданной строки s. Если таких несколько, можно вывести любую из них.

Формат входного файла

В первой строке входного файла содержится единственное целое число n ($1 \le n \le 200$) — длина строки s. Вторая строка содержит s. Строка s состоит из строчных букв латинского алфавита

Формат выходного файла

Выведите наидлиннейшую 2-подстроку заданной строки s. Если таких несколько, выведите любую из них.

Примеры

input.txt	output.txt
7 cabacaa	acaa
3 aaa	aaa
6 abcabc	bc

Задача B. Дисплей

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Монохромный дисплей монитора имеет разрешение n пикселей по вертикали и m — по горизонтали. Каждый пиксель на дисплее имеет либо черный, либо белый цвет. Состояние дисплея задано в виде матрицы $n \times m$, где n — количество строк, а m — столбцов. Элемент матрицы равен 0, если соответствующий пиксель дисплея черный, и 1 — если белый.

Битым называется такой белый пиксель, все соседние по стороне пиксели с которым — черные. В общем случае, у пикселя четыре соседних по стороне, но может быть меньше, если он расположен на границе или в углу дисплея.

Найдите количество квадратов 3×3 из пикселей, которые содержат хотя бы один битый пиксель.

Формат входного файла

Первая строка входного файла содержит записанные через пробел целые числа n и m ($3 \le n, m \le 400$). Далее следует описание дисплея в виде последовательности из n строк. Каждая строка содержит m записанных через пробел целых чисел — цвета пикселей в соответствующей строке дисплея. Число 0 обозначает черный цвет пикселя, а 1 — белый.

Формат выходного файла

Выведите искомое количество.

Примеры

input.txt	output.txt
3 7 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0	3
4 4 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0	4

Примечание

В первом примере битыми являются два пикселя — самый левый белый пиксель и самый правый белый пиксель. Существует один 3×3 квадрат, содержащий первый из них, и два — второй.

Во втором примере все белые пиксели являются битыми, следовательно, каждый из четырех возможных 3×3 квадратов содержит хотя бы один битый пиксель.

Задача C. Чат

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам заданы записи журнала работы некоторого простого чата. Записи заданы в хронологическом порядке. В журнале встречаются строки трех возможных типов:

- «+u» — обозначает, что пользователь u зашел в чат (например, «`+mike`»);
- «-u» — обозначает, что пользователь u покинул чат (например, «`-mike`»);
- «u_1:u_2»— обозначает, что пользователь u_1 написал сообщение пользователю u_2 ($u_1 \neq u_2$).

Гарантируется, что записи в журнале работы чата правильные, а именно:

- не встречается строка «+u», если пользователь u уже в чате в данный момент;
- не встречается строка «-u», если пользователь u отсутствует в чате в данный момент;
- если встречается строка «u_1:u_2», то гарантируется, что пользователь u_1 находится в данный момент в чате, $u_1 \neq u_2$.

В том случае, если пользователь u_1 посылает сообщение пользователю u_2, а тот отсутствует в чате в момент отсылки сообщения, то сообщение чатом полностью игнорируется. Оно не будет доставлено адресату, даже если u_2 позже появится в чате. Пользователь не может отослать сообщение самому себе.

Назовем пару пользователей a и b друзьями, если a написал сообщение b (и оно было доставлено), а также b написал сообщение a (и оно тоже было доставлено). Порядок, в котором пользователи обменялись сообщениями не имеет значения. Конечно, пара друзей (a, b) и пара (b, a) — это одна и та же пара.

По содержимому журнала работы чата найдите количество пар друзей среди пользователей.

Формат входного файла

Первая строка содержит целое число n ($1 \leq n \leq 50\,000$) — количество строк в журнале работы чата. Далее n строк входного файла содержат записи в журнале. Записи даны в порядке их появления, начиная с первой записи от начала работы чата. Изначально, в чате пользователей нет. Каждая запись задана на отдельной строке, имя каждого пользователя состоит из строчных букв латинского алфавита и имеет длину от 1 до 10 символов включительно.

Формат выходного файла

Выведите искомое количество.

Примеры входных и выходных данных приведены на следующей странице.

Примеры

input.txt	output.txt
10 +mike +peter mike:peter -mike peter:mike +ivan ivan:peter peter:ivan -peter ivan:mike	1
8 +tanya +helen +katya tanya:helen tanya:katya helen:katya helen:tanya katya:tanya	2

Примечание

В первом примере друзьями являются `ivan` и `peter`, а во втором две пары друзей — `tanya`, `helen` и `tanya`, `katya`.

Задача D. Обработка XML

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

XML — текстовый формат, предназначенный для хранения структурированных данных. В этой задаче используются лишь небольшая часть возможностей XML, многие особенности этого формата упомянуты не будут, а в тексте задачи будет дано упрощенное понимание XML.

Документы XML записывается с помощью тегов, которые бывают двух видов — открывающие и закрывающие:

- открывающие теги записываются в форме `<тег>`, например, `<users>`;
- закрывающие теги записываются в форме `</тег>`, например, `</users>`.

Открывающие и закрывающие теги образуют правильную скобочную последовательность. Пары тегов — открывающий тег и парный ему закрывающий, называются элементами. Если элемент не содержит внутри себя других данных, то открывающий тег и закрывающий могут быть совмещены в один, например, `<friends/>`.

Теги могут иметь атрибуты, но в этой задаче возможный атрибут только один `name` и он обязательно присутствует у открывающих тегов `user` и `friend`, у других тегов атрибутов нет. Пример записи открывающих тегов с атрибутом `name`: `<user name="andrey">` или `<friend name="helen">`.

Для структурированной записи информации о пользователях некоторой простой социальной сети используется XML определенного вида. Заданный XML-документ состоит из одного элемента `users`, который содержит один или более элемент `user`. Каждый элемент `user` содержит ровно один элемент `friends`. Каждый элемент `friends` содержит ноль или более элементов `friend`. Элементы `user` и `friend` содержат атрибут `name`, значение которого являтся строкой строчных букв латинского алфавита длины от 1 до 10 символов включительно. Вот пример возможного XML-документа:

```
<users>
<user name="mike">
<friends>
    <friend name="vasya"/><friend name="peter"/>
</friends>
</user>
<user name="peter">
<friends>
   </friends>
</user>
<user name="mike"><friends><friend name="peter"/></friends></user>
</users>
```

Этот документ описывает, что в системе есть два пользователя — `mike` и `peter`. У пользователя `mike` есть два друга — `vasya` и `peter`, а у `peter` — друзей нет. Заметьте, что документ дважды описывает пользователя `mike`, а пользователя `vasya` в системе нет.

Данные в XML-документе могут оказаться не до конца корректны. Ваша задача состоит в том, чтобы исправить XML-документ, по следующим правилам:

- каждый элемент `friend` со ссылкой на друга, который не является пользователем системы должен быть удален (например, элемент `<friend name="vasya"/>` из примера выше);
- пользователь может быть описан в XML более одного раза, тогда необходимо слить списки друзей в таких описаниях и оставить только одно;
- в списках друзей могут быть дубликаты, их необходимо удалить (дубликаты могут появиться после слияния, их тоже надо удалять).
- если пользователь содержит самого себя в качестве друга, то соответствующий элемент `friend` надо удалить.

После преобразования заданного XML выведите его, упорядочив элементы `user` лексикографически по именам пользователей (значению атрибута `name`). Списки друзей каждого пользователя упорядочивайте аналогичным образом.

Формат входного файла

Входной файл содержит документ XML, описанного вида. Теги могут разделяться произвольным количеством пробелов, переводами строк и т.п. Лишние пробелы внутри тегов (между символами '<' и '>') не встречаются, пробел используется в единственном случае — перед записью атрибута (между именем тега и атрибутом используется ровно один пробел). Размер входного файла не превосходит 1 мегабайт. Для записи значения атрибута `name` используются двойные кавычки.

Формат выходного файла

Выведите обработанный XML-документ. Строго следуйте всем правилам обработки.

Выходной документ **не должен** содержать средства XML, не описанные в условии задачи (такие как, инструкции по обработке, DTD, сущности и ссылки на них или другие элементы). Проверка правильности вывода будет производиться с точностью до пробелов, табуляций и переводов строк, следовательно, вы можете форматировать вывод пробелами, табуляциями и переводами строк произвольным образом. Пустые теги можно выводить как в развернутой, так и сокращенной форме. Разрешается использовать как двойные, так и одинарные кавычки (апострофы) для обрамления значений атрибутов (т.е. символы с ASCII кодами 34 и 39).

Примеры входных и выходных данных приведены на следующей странице.

Примеры

input.txt

```
<users>
<user name="mike">
<friends>
    <friend name="vasya"/><friend name="peter"/>
</friends>
</user>
<user name="peter">
<friends>
   </friends>
</user>
<user name="mike"><friends><friend name="peter"/></friends></user>
</users>
```

output.txt

```
<users>
<user name="mike">
<friends>
    <friend name="peter"/>
</friends>
</user>
<user    name="peter">
    <friends/>
</user>
</users>
```