Задача A. Разбор утверждения

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

На вход вашей программе дается утверждение в следующей грамматике:

⟨Файл⟩ ::= ⟨Выражение⟩
⟨Выражение⟩ ::= ⟨Дизъюнкция⟩ | ⟨Дизъюнкция⟩ '->' ⟨Выражение⟩
⟨Дизъюнкция⟩ ::= ⟨Конъюнкция⟩ | ⟨Дизъюнкция⟩ '|' ⟨Конъюнкция⟩
⟨Конъюнкция⟩ ::= ⟨Отрицание⟩ | ⟨Конъюнкция⟩ '&' ⟨Отрицание⟩
⟨Отрицание⟩ ::= '!' ⟨Отрицание⟩ | ⟨Переменная⟩ | '(' ⟨Выражение⟩ ')'
⟨Переменная⟩ ::= ('A'...'Z') {'A'...'Z' | '0'...'9' | ''}*

Имена переменных не содержат пробелов. Между символами оператора '->' нет пробелов. В остальных местах пробелы могут присутствовать. Символы табуляции и возврата каретки должны трактоваться как пробелы.

Вам требуется написать программу, разбирающую утверждение и строящую его дерево разбора, и выводящую полученное дерево в единственной строке без пробелов в следующей грамматике:

⟨Файл⟩ ::= ⟨Вершина⟩
⟨Вершина⟩ ::= '(' ⟨Знак⟩ ',' ⟨Вершина⟩ ',' ⟨Вершина⟩ ')'
| '(!'⟨Вершина⟩')'
| ⟨Переменная⟩
⟨Знак⟩ ::= '->' | '|' | '&'
⟨Переменная⟩ ::= ('A'...'Z') {'A'...'Z' | '0'...'9' | ''}*

Формат входных данных

В единственной строке входного файла дано утверждение в грамматике из условия. Размер входного файла не превышает 100 КБ.

Формат выходных данных

В единственной строке выходного файла выведите дерево разбора утверждения без пробелов.

Примеры

стандартный ввод
`!A&!B->!(A|B)`
стандартный вывод
`(->,(&,(!A),(!B)),(!(|,A,B)))`

стандартный ввод
`P1'->!QQ->!R10&S|!T&U&V`
стандартный вывод
`(->,P1',(->,(!QQ),(|,(&,(!R10),S),(&,(&,(!T),U),V))))`

Задача B. Минимизация доказательства

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	15 секунд
Ограничение по памяти:	512 мегабайт

На вход вашей программе дается доказательство утверждения в следующей грамматике:

⟨Файл⟩ ::= ⟨Контекст⟩ '|-' ⟨Выражение⟩ '\n' ⟨Строка⟩*
⟨Контекст⟩ ::= ⟨Выражение⟩ [',' ⟨Выражение⟩]*
| ''
⟨Строка⟩ ::= ⟨Выражение⟩ '\n'
⟨Выражение⟩ ::= ⟨Выражение⟩ '&' ⟨Выражение⟩
| ⟨Выражение⟩ '|' ⟨Выражение⟩
| ⟨Выражение⟩ '->' ⟨Выражение⟩
| '!' ⟨Выражение⟩
| '(' ⟨Выражение⟩ ')'
| ⟨Переменная⟩
⟨Переменная⟩ ::= ('A'...'Z') {'A'...'Z' | '0'...'9' | ''}*

Операторы '&' и '|' левоассоциативны. Оператор '->' правоассоциативен. Операторы в порядке уменьшения приоритета: '!', '&', '|', '->'.

Имена переменных не содержат пробелов. Между символами одного оператора нет пробелов ('->' и '|-'). В остальных местах пробелы могут присутствовать. Символы табуляции и возврата каретки должны трактоваться как пробелы.

Требуется проверить доказательство на корректность. Если оно неверно, выведите «`Proof is incorrect`». Иначе минимизируйте и проаннотируйте доказательство.

Под минимизацией доказательства подразумевается создание нового доказательства такого, что:

- Новое доказательство доказывает то же самое утверждение в том же самом контексте
- Строки нового доказательства являются подпоследовательностью строк исходного доказательства
- В новом доказательстве ни одно выражение не встречается в нескольких строках
- В новом доказательстве нет неиспользуемых выражений, т.е. все выражения, кроме последнего, должны использоваться одним или более применением правила Modus Ponens.

Под аннотированием доказательства подразумевается:

- Все строки должны быть пронумерованы
- Каждая строка должна содержать пояснение, как она была выведена:
  1. Аксиома: номер аксиомы
  2. Предположение: номер предположения
  3. Modus Ponens: номера строк, в которых записаны выражения, используемые для вывода выражения в текущей строке

Формат входных данных

Во входном файле задано доказательство в приведенной выше грамматике. Размер входного файла не превышает 10 МБ.

Формат выходных данных

Если данное доказательство является некорректным, в единственной строке выходного файла должна быть запись «`Proof is incorrect`».

Иначе в файле должно быть минимизированное проаннотированое корректное доказательство. Каждая строка, кроме последней, должна быть использована хотя бы в одной аннотации Modus Ponens. Подробный формат аннотаций смотрите в примерах.

Примеры

стандартный ввод

```
|- A -> A
A & A -> A
A -> A -> A
A -> (A -> A) -> A
A & A -> A
(A -> A -> A) -> (A -> (A -> A) -> A) -> A -> A
(A -> (A -> A) -> A) -> A -> A
A & A -> A
A -> A
```

стандартный вывод

```
|- (A -> A)
[1. Ax. sch. 1] (A -> (A -> A))
[2. Ax. sch. 1] (A -> ((A -> A) -> A))
[3. Ax. sch. 2] ((A -> (A -> A)) -> ((A -> ((A -> A) -> A)) -> (A -> A)))
[4. M.P. 3, 1] ((A -> ((A -> A) -> A)) -> (A -> A))
[5. M.P. 4, 2] (A -> A)
```

стандартный ввод

```
A->B, !B |- !A
A->B
!B
!B -> A -> !B
A -> !B
(A -> B) -> (A -> !B) -> !A
(A -> !B) -> !A
!A
```

стандартный вывод

```
(A -> B), !B |- !A
[1. Hypothesis 1] (A -> B)
[2. Hypothesis 2] !B
[3. Ax. sch. 1] (!B -> (A -> !B))
[4. M.P. 3, 2] (A -> !B)
[5. Ax. sch. 9] ((A -> B) -> ((A -> !B) -> !A))
[6. M.P. 5, 1] ((A -> !B) -> !A)
[7. M.P. 6, 4] !A
```

стандартный ввод

```
A, C |- B'
B'
```

стандартный вывод

```
Proof is incorrect
```

Задача C. Теорема Гливенко

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	15 секунд
Ограничение по памяти:	512 мегабайт

На вход вашей программе дается **корректное** доказательство утверждения α в классическом исчислении высказываний. Доказательство записано с использованием грамматики из предыдущего задания.

Вам требуется построить корректное доказательство утверждения $\neg\neg\alpha$ в интуиционистском исчислении высказываний.

Формат входных данных

Во входном файле задано доказательство утверждения α в классическом исчислении высказываний. Размер входного файла не превышает 5 КБ.

Формат выходных данных

Файл должен содержать корректное доказательство утверждения $\neg\neg\alpha$ в интуиционистском исчислении высказываний в том же контексте, что доказательство α во входном файле.

Пример

стандартный ввод
`A |- A` `A`
стандартный вывод

`A |- !!A`
`A`
`(A -> (!A -> A))`
`(!A -> A)`
`(!A -> (!A -> !A))`
`((!A -> (!A -> !A)) -> ((!A -> ((!A -> !A) -> !A)) -> (!A -> !A)))`
`((!A -> ((!A -> !A) -> !A)) -> (!A -> !A))`
`(!A -> ((!A -> !A) -> !A))`
`(!A -> !A)`
`((!A -> A) -> ((!A -> !A) -> !!A))`
`((!A -> !A) -> !!A)`
`!!A`

Замечание

В классическом исчислении высказываний используются следующие схемы аксиом:

(1) $\alpha \to \beta \to \alpha$
(2) $(\alpha \to \beta) \to (\alpha \to \beta \to \gamma) \to (\alpha \to \gamma)$
(3) $\alpha \to \beta \to \alpha\&\beta$
(4) $\alpha\&\beta \to \alpha$
(5) $\alpha\&\beta \to \beta$
(6) $\alpha \to \alpha \vee \beta$
(7) $\beta \to \alpha \vee \beta$
(8) $(\alpha \to \gamma) \to (\beta \to \gamma) \to (\alpha \vee \beta \to \gamma)$
(9) $(\alpha \to \beta) \to (\alpha \to \neg\beta) \to \neg\alpha$
(10) $\neg\neg\alpha \to \alpha$

В интуиционистском исчислении высказываний 10-я схема аксиом заменяется на:

(10) $\alpha \to \neg\alpha \to \beta$

Задача D. Полнота исчисления высказываний

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	15 секунд
Ограничение по памяти:	512 мегабайт

На вход вашей программе дается утверждение α в грамматике из предыдущих заданий.
От вас требуется найти:

- Набор гипотез Γ_1 со следующими свойствами:
  - Γ_1 состоит только из переменных
  - $\Gamma_1 \vdash \alpha$

  В этом случае вам нужно вывести доказательство $\Gamma_1 \vdash \alpha$.
- Если такого набора гипотез не нашлось, то нужно найти наименьший набор гипотез Γ_2:
  - Γ_2 состоит только из отрицаний переменных
  - $\Gamma_2 \vdash \neg\alpha$

  В этом случае вам нужно вывести доказательство $\Gamma_2 \vdash \neg\alpha$.
- Если и такого набора гипотез не нашлось, то выведите «`:(`».

Если среди предыдущих случаев существует несколько подходящих наборов гипотез (а если такие наборы есть, то их всегда бесконечно много), то требуется вывести любой подходящий набор наименьшего размера.

Формат входных данных

Во входном файле задано утверждение α. Размер входного файла не превышает 50 байт. Количество различных переменных, входящих в α, не превосходит 3.

Формат выходных данных

Если требуемого набора гипотез не существует, в единственной строке выведите «`:(`». Иначе выведите требуемое в условии доказательство, используя грамматику из предыдущих заданий.

Примеры

стандартный ввод
`!A`
стандартный вывод
`:(`

стандартный ввод
`A -> A & B`
стандартный вывод

`B |- A -> A & B`
`B`
`B -> A -> B`
`A -> B`
`A -> B -> A & B`
`(A -> B) -> (A -> B -> A & B) -> A -> A & B`
`(A -> B -> A & B) -> A -> A & B`
`A -> A & B`

Задача E. Проверка доказательства в формальной арифметике

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напишите программу, проверяющую доказательство в формальной арифметике на корректность. Докзательство соответствует следующей грамматике.

⟨ввод⟩ ::= ⟨заголовок⟩ '`\n`' ⟨доказательство⟩
⟨заголовок⟩ ::= [{⟨выражение⟩'`,`'}* ⟨выражение⟩] '`|-`' ⟨выражение⟩
⟨доказательство⟩ ::= {⟨выражение⟩'`\n`'}*
⟨выражение⟩ ::= ⟨дизъюнкция⟩ | ⟨дизъюнкция⟩ '`->`' ⟨выражение⟩
⟨дизъюнкция⟩ ::= ⟨конъюнкция⟩ | ⟨дизъюнкция⟩ '`|`' ⟨конъюнкция⟩
⟨конъюнкция⟩ ::= ⟨унарное⟩ | ⟨конъюнкция⟩ '`&`' ⟨унарное⟩
⟨унарное⟩ ::= ⟨предикат⟩ | '`!`' ⟨унарное⟩ | '`(`' ⟨выражение⟩ '`)`'
| ('`@`'|'`?`') ⟨переменная⟩ '`.`' ⟨выражение⟩
⟨переменная⟩ ::= ('`a`' ... '`z`') {'`0`'... '`9`'}*
⟨предикат⟩ ::= ('`A`' ... '`Z`') {'`0`'... '`9`'}* ['`(`' ⟨терм⟩ { '`,`' ⟨терм⟩ }* '`)`']
| ⟨терм⟩ '`=`' ⟨терм⟩
⟨терм⟩ ::= ⟨слагаемое⟩ | ⟨терм⟩ '`+`' ⟨слагаемое⟩
⟨слагаемое⟩ ::= ⟨умножаемое⟩ | ⟨слагаемое⟩ '`*`' ⟨умножаемое⟩
⟨умножаемое⟩ ::= ('`a`' ... '`z`') {'`0`'... '`9`'}* '`(`' ⟨терм⟩ { '`,`' ⟨терм⟩ }* '`)`'
| ⟨переменная⟩ | '`(`' ⟨терм⟩ '`)`'
| '`0`' | ⟨умножаемое⟩ '`'`'

Формат входных данных

Доказательство.

Формат выходных данных

Если доказательство корректно, выведите «`Proof is correct`». Если в доказательстве нет ошибок, но требуемое выражение не доказано, выведите «`Required hasn't been proven`». Иначе, выведите «`Line #x can't be obtained`», где x — номер первого некорректной строки в доказательстве. Строки доказательства нумеруются с 1.

Задача F. Представимость в формальной арифметике

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напомним, что числа a и b называются сравнимыми по модулю 2, если они имеют одинаковый остаток при делении на 2 — то есть, они либо одновременно чётные, либо одновременно нечётные. Построим соответствующее отношение $R = \{(a, b) \mid a \equiv b \pmod 2\}$.

Покажите, что отношение R представимо в формальной арифметике с помощью формулы ρ:

$$\rho(x, y) \coloneqq \exists p.\exists q.(2p = x \wedge 2q = y) \vee (2p = x + 1 \wedge 2q = y + 1)$$

То есть, напишите программу, которая по натуральным числам a, b ($a, b \in \mathbb{N}_0$) построит доказательство:

- $\vdash \rho[x \coloneqq \overline{a}, y \coloneqq \overline{b}]$, если $a \equiv b \pmod 2$;
- $\vdash \neg\rho[x \coloneqq \overline{a}, y \coloneqq \overline{b}]$, если $a \not\equiv b \pmod 2$.

Формат входных данных

В единственной строке даны два целых числа a и b ($0 \leqslant a, b < 100$).

Формат выходных данных

Выведите доказательство в формате, описанном в предыдущем задании.