Задача A. Истинность высказываний

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

Как известно, формула классического исчисления высказываний может быть:

1. невыполнимой — нет значений переменных, при которых формула истинна;
2. одновременно выполнимой и опровержимой — то есть, существуют такие наборы значений переменных, что формула истинна (формула выполнима), и такие значения, что формула ложна (формула к тому же ещё и опровержима);
3. общезначимой — при всех значениях переменных формула истинна.

По данной на вход формуле определите, какой из вариантов имеет место.

Формат входных данных

На вход подаётся формула классического исчисления высказываний с не более чем 16 различными переменными, состоящая из не более чем 256 символов, заданная в грамматике из первой задачи.

Формат выходных данных

Выведите одну из трёх строк:

1. `Unsatisfiable` — если формула невыполнима.
2. `Satisfiable and invalid,` t `true and` f `false cases` — если формула выполнима и опровержима. Числа t и f обозначают количество наборов значений переменных, при которых формула истинна (t) и ложна (f).
3. `Valid` — если формула общезначима.

Примеры

стандартный ввод
`A&!A`
стандартный вывод
`Unsatisfiable`

стандартный ввод
`A->!B123`
стандартный вывод
`Satisfiable and invalid, 3 true and 1 false cases`

стандартный ввод
`((PPP->PPP')->PPP)->PPP`
стандартный вывод
`Valid`

Замечание

Построим таблицу истинности для формулы $A \to \neg B123$:

A	B	$\neg B123$	$A \to \neg B123$
Л	Л	И	И
Л	И	Л	И
И	Л	И	И
И	И	Л	Л

Подсчитаем: формула истинна на трёх наборах значений переменных, и ложна на одном наборе.

Задача B. Теорема о дедукции

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	15 секунд
Ограничение по памяти:	1024 мегабайта

На вход вашей программе даётся корректное доказательство высказывания β в контексте $\gamma_1, \gamma_2, \ldots, \gamma_n, \alpha$. Требуется перестроить его в доказательство $\gamma_1, \gamma_2, \ldots, \gamma_n \vdash \alpha \to \beta$.

Формат входных данных

На вход дается доказательство утверждения в соответствии со следующей грамматикой:

⟨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: '`!`', '`&`', '`|`', '`->`'. Коды символов: код апострофа ('`'`') — 39, код вертикальной черты ('`|`') — 124.

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.

Формат выходных данных

Формат выходного файла совпадает с форматом входного файла.

Примеры

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`

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

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

стандартный ввод
A|-A->A A->A->A 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 (A -> A) -> ((A -> (A -> (A -> A))) -> (A -> (A -> A))) (A -> (A -> (A -> A))) -> (A -> (A -> A)) A -> (A -> A)

Задача C. Формальная арифметика 2022

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.

Формат входных данных

⟨Файл⟩	::=	⟨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`’
⟨предикат⟩	::=	‘`A`’ . . . ‘`Z`’
	|	⟨терм⟩‘`=`’⟨терм⟩
⟨терм⟩	::=	⟨слагаемое⟩ | ⟨терм⟩‘`+`’⟨слагаемое⟩
⟨слагаемое⟩	::=	⟨умножаемое⟩ | ⟨слагаемое⟩‘`*`’⟨умножаемое⟩
⟨умножаемое⟩	::=	⟨переменная⟩ | ‘`(`’⟨терм⟩‘`)`’
	|	‘`0`’ | ⟨умножаемое⟩‘`’`’

Коды символов: символ апострофа (’) — 0x27, вертикальная черта (|) — 0x7c.

Формат выходных данных

Если доказательство корректно, проаннотируйте его. Первая строка должна повторять строку из входного файла, остальные строки доказательства должны быть предварены аннотацией:

- `[n. Hyp. k]`, где n — номер выражения, а k — номер гипотезы.
- `[n. Ax. sch. k]`, где k — номер схемы аксиом: либо число от 1 до 12, либо $A9$.
- `[n. Ax. k]`, где k — значение от $A1$ до $A8$.
- `[n. M.P. k, l]`, `[n. ?-rule k]`, `[n. @-rule k]` — для правил вывода. Смысл индексов для M.P.: если доказательство представлено формулами δ_i, то запись слева означает $\delta_l \equiv \delta_k \to \delta_n$.

Аннотации перечислены в порядке предпочтения: если выражение может быть обосновано, допустим, как аксиома $A8$ или как $M.P.$, в ответе должно быть указано `Ax. A8`. В случае пересечения аксиом/схем указывайте аксиому/схему с минимальным номером; арифметические аксиомы/схемы идут после логических. Аксиомы упорядочены в соответствии с номерами, вводившимися на лекции. Если выражение может быть получено при помощи одного правила вывода несколькими способами, предпочтение должно отдаваться наиболее ранним ссылкам в лексикографическом порядке: `M.P. 1,10` предпочтительнее `M.P. 10,1`. Modus Ponens предпочтительнее правил с кванторами, правило с квантором существования предпочтительнее правила с квантором всеобщности (даже если номер исходной формулы для правила с квантором существования меньше). Также, аксиомы предпочтительнее правил вывода.

В выражениях должны быть расставлены все скобки в точности по одному разу (т.е. скобки вокруг всех унарных и бинарных выражений — кроме апострофов).

Если доказательство некорректно, выведите одну из следующих строк, в зависимости от типа ошибки. Ваша программа должна находить первое некорректное выражение в доказательстве, и

для него указывать тип ошибки с минимальным номером (в соответствии со списком ниже). Не забывайте, что строка, некорректная с точки зрения одной из аксиом (правила вывода), может тем не менее, быть корректна с точки зрения другого правила.

1. `Expression n: variable v occurs free in ?-rule.`
2. `Expression n: variable v occurs free in @-rule.`
3. `Expression n: variable v is not free for term t in ?-axiom.`
4. `Expression n: variable v is not free for term t in @-axiom.`
5. `Expression n is not proved.`
6. `The proof proves different expression.`

Все строки доказательства, предшествующие некорректной, должны быть проаннотированы.

Столь подробные правила введены для того, чтобы упростить проверяющую программу: ответ сравнивается с эталонным на равенство; будьте внимательны.

Пример

стандартный ввод

`|-a+0=a`
`(((a)+0))=a`
`(@y.y+0*0'=y)->(?x.@y.x=y)`

стандартный вывод

`|-((a+0)=a)`
`[1. Ax. A5] ((a+0)=a)`
`Expression 2: variable x is not free for term (y+(0*0')) in ?-axiom.`