General information

- You have three hours to implement the solution of this task. The timer starts after questions and answers regarding the task. You are allowed to ask questions once the test commences, but do that without disturbing your colleagues work.
- Your solution won't be tested in an automatic manner, but rather manually. It is not necessary to 100% comply with the assignment (e.g. the content of error/information messages). You will be evaluated based on the implementation and the functionality of main concepts of the assignment.
- It is allowed to use both paper and electronic study materials, such as slides from the courses, if it doesn't include the solution to this task. Any means of communication with anyone for any reason is prohibited.
- If you manage to complete the solution before the deadline, it is allowed to have it checked immediately. If the solution still has some bugs, you're allowed to fix them within the time limit.
- After successfully passing the test, send all source files via email with subject `Test 14.1.2020` to the examiner — `kliber@d3s.mff.cuni.cz`.

Assignment — Checking of assignments of types in type system with hierarchy

Your task is to develop an application, which loads definition of simple type system and then will answer the questions related to possible assignments between types in this system.

If the program is started with different number of arguments on the command line then 1, it will output following text to the standard output `Usage: Program.exe typesFile` and will terminate. Otherwise, it will work in two phases. First it will process declaration of types, and if this phase finishes successfully, the application will then allow its user to query the type system.

Input file format — Declaration of types

The program will process the only file passed on the command line. This file contains one declaration per line which can be in one of two formats:

Name `:` *Parent*	Deklaruje typ *Name*, který dědí od typu *Parent*.
Name	Deklaruje typ *Name*, který dědí od typu `Object`

In order for the type to be declared, the following must hold:

- This type was not yet declared. Otherwise the following error will be printed: *N*`: Duplicate type` *X*`. First declared on line` *M*, where *N* is the line number where the error occurred, *M* is the line number where the first was declared for the first time and *X* is the name of the type.
- The parent of the type must be already declared. Otherwise the following error will be printed: *N*`: Non existing base type` *X*, where *N* is the line number where the error occurred and *X* is the name of the type.
- A special type `Object` is always declared on zero-th line and has no parent (this is the only type with such property).

If any of described error occurs while processing the input file, then such line doesn't declare any type, but the processing goes on. If the file is processed without any error then the next phase will take place. Otherwise, the following error shall be printed: *K* `errors. Stopping.`, where *K* is the total number of errors.

Queries format

The program will read queries from the standard input which are immediately processed. The query has the following format:

`A = B_n B_n-1 ... B_1`

Where $n \geq 1$ and answer to the query can be one of the following:

- `Assignment A = B_1 is valid`, if it is possibel to cast type B_1 to the type B_2, ..., type B_{n-1} to the type B_n and type B_n can be assigned to the type A.
- `Assignment A = B_1 is invalid`, if one of the earlier mentioned steps fails.
- `Cannot find type X`, if any of types A, B_1, ..., B_n wasn't included in the input file, where X is the name of such non-declared type.

Assignment from the type `Y` to the type `X` (i.e. `X = Y`) is valid, iff `X` is effectively `Y` or `X` is the ancestor of `Y` (to any level)

Cast of the type `X` to the type `Y` (i.e. `Y X`) is valid, iff `X` is effectively `Y` or if `Y` can be assigned to `X` (`X = Y` is valid) or if `X` is some descendant of `Y` (to any level).

Restrictions

- You may assume, that the input file is always in the correct format (syntactically), according to this assignment. Although it may happen that a logical error occurs during the declaration of the type.
- You may assume, that queries always are in the format according to this assignment. One can separate queries by newline which should be ignored by the application.

Examples

Lines starting with dollar sign (`$`) represent invoking the application. Lines starting with greater than sign (`>`) represent application's standard output. Lines starting with less than sign (`<`) represent application's standard input. None of these characters are part of the output of the program and are used only for clarification. Files for testing and electronic version is available on `http://d3s.mff.cuni.cz/~kliber/test001.zip`.

```
$ Program.exe
> Usage: Program.exe typesfile
```

```
$ Program.exe data.invalid
> 1: Duplicate type Object. First declared on line 0
> 2: Non existing base type A.
> 3: Non existing base type A.
> 6: Duplicate type D. First declared on line 5
> 4 errors. Stopping.
```

```
$ cat data.invalid
> Object
> A : A
> B : A
> C
> D
> D : C
```

```
$ Program.exe data
< Object = Base
> Assignment Object = Base is valid
< Base = Derived
> Assignment Base = Derived is valid
< Object = Derived
> Assignment Object = Derived is valid
< Base = Object
> Assignment Base = Object is invalid
< Base = Base Object
> Assignment Base = Object is valid
```

```
$ cat data
> Base
> Derived : Base
```