Problem H. Hidden Hierarchy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are working on the user interface for a simple text-based file explorer. One of your tasks is to build a navigation pane displaying the *directory hierarchy*. As usual, the filesystem consists of directories which may contain files and other directories, which may, in turn, again contain files and other directories etc. Hence, the directories form a hierarchical tree structure. The top-most directory in the hierarchy is called the *root* directory. If directory d directly contains directory e we will say that d is the *parent directory* of e while e is a *subdirectory* od d. Each file has a *size* expressed in bytes. The directory size is simply the total size of all files directly or indirectly contained inside that directory.

All files and all directories except the root directory have a *name* — a string that always starts with a letter and consists of only lowercase letters and "." (dot) characters. All items (files and directories) directly inside the same parent directory must have unique names. Each item (file and directory) can be uniquely described by its *path* — a string built according to the following rules:

- Path of the root directory is simply "/" (forward slash).
- For a directory d, its path is obtained by concatenating the directory names top to bottom along the hierarchy from the root directory to d, preceding each name with the "/" character and placing another "/" character at the end of the path.
- For a file f, its path is the concatenation of the parent directory path and the name of file f.

We display the directory hierarchy by *printing* the root directory. We print a directory d by outputting a line of the form "$m_d \_ p_d \_ s_d$" where p_d and s_d are the path and size of directory d respectively, while m_d is its *expansion marker* explained shortly. If d contains other directories we must choose either to *collapse* it or to *expand* it. If we choose to expand d we print (using the same rules) all of its subdirectories in lexicographical order by name. If we choose to collapse directory d, we simply ignore its contents.

The expansion marker m_d is a single blank character when d does not have any subdirectories, "+" (plus) character when we choose to collapse d or a "-" (minus) character when we choose expand d.

Given a list of files in the filesystem and a threshold integer t, display the directory hierarchy ensuring that each directory of size at least t is printed. Additionally, the total number of directories printed should be minimal. Assume there are no empty directories in the filesystem — the entire hierarchy can be deduced from the provided file paths. Note that the root directory has to be printed regardless of its size. Also note that a directory of size at least t only has to be *printed*, but not necessarily *expanded*.

Input

The first line contains an integer n ($1 \le n \le 1\,000$) — the number of files. Each of the following n lines contains a string f and an integer s ($1 \le s \le 10^6$) — the path and the size of a single file. Each path is at most 100 characters long and is a valid file path according to the rules above. All paths will be different.

The following line contains an integer t ($1 \le t \le 10^9$) — the threshold directory size.

Output

Output the minimal display of the filesystem hierarchy for the given threshold as described above.

Example

standard input	standard output
9 /sys/kernel/notes 100 /cerc/problems/a/testdata/in 1000000 /cerc/problems/a/testdata/out 8 /cerc/problems/a/luka.cc 500 /cerc/problems/a/zuza.cc 5000 /cerc/problems/b/testdata/in 15 /cerc/problems/b/testdata/out 4 /cerc/problems/b/kale.cc 100 /cerc/documents/rules.pdf 4000 10000	- / 1009727 - /cerc/ 1009627 /cerc/documents/ 4000 - /cerc/problems/ 1005627 - /cerc/problems/a/ 1005508 /cerc/problems/a/testdata/ 1000008 + /cerc/problems/b/ 119 + /sys/ 100
8 /b/test/in.a 100 /b/test/in.b 1 /c/test/in.a 100 /c/test/in.b 1 /c/test/pic/in.a.svg 10 /c/test/pic/in.b.svg 10 /a/test/in.a 99 /a/test/in.b 1 101	- / 322 + /a/ 100 - /b/ 101 /b/test/ 101 - /c/ 121 + /c/test/ 121
2 /a/a/a 100 /b.txt 99 200	+ / 199