CMPUT 275 - Tangible Computing
Interview Problem: Bash History

Description

You are given a simplified bash history file that consists entirely of the following commands:

- `cd <directoryname>`
- `cd ..`
- `ls <filename>`

Your task is to summarize all directories and files that were seen in this bash history. You may assume that all listed commands run successfully.

Note, there are no hard or symbolic links in the directory structure (if you do not know about these, then don't worry about it).

You may find the map data structure useful, it functions similarly to an `unordered_map`, but all keys will be stored in lexicographically sorted order. The requirements for using such a map is that the items are comparable via the `<` operator.

You may also find the set data structure useful, which is an ordered set: items will be enumerated in sorted order when you iterate through a set. In general, you are permitted to use anything from the C++ standard library.

Input

The first line of input will be an integer $1 \leq n \leq 100,000$, the number of commands in the input.

Then follows n lines, each containing one of the aforementioned commands. You may assume every line is of this form: no error checking is required. There will be a single space separating the first string (either `ls` or `cd`) and the second string on each of these n lines.

Every file or directory name appearing on a line in the input will consist of between 1 and 10 characters. All characters in a file or directory name will be a letter (lowercase or uppercase) or a digit. There will be no spaces in any file or directory name.

The command `cd ..` will not appear when we are currently in the root directory. The maximum depth of any directory is 20, meaning at any point it would take at most 20 `cd ..` commands to go back to the root directory.

Output

Output should contain one line for each unique directory, in the format:
<*pathname*> <$filename_1$> <$filename_2$> ... <$filename_k$>

Here, <*pathname*> refers to the full path of the directory from the root and would appear exactly as it would if you used the command `pwd` from within that directory (see the sample output below).

The lines should appear in lexicographic order of the full path name. For each line, the files within that directory should be output in lexicographic order. Here, by lexicographic order we simply mean by using `<` comparison with C++ strings.

e.g. If `/b` contains the files `hello` and `world` and `/a` contains the files `bash`, `is` and `cool`, the expected output would be:

```
/a bash cool is
/b hello world
```

If output is not lexicographically sorted you will receive zero credit for correctness.

Be careful to avoid extra spaces at the end of a line of output.

Running Time

There is no specific $O()$ running time you must achieve on this weekly exercise. You simply have to pass the test cases. We have included an example of the largest-possible size in the Test Center files on eClass. An instructor solution that is a bit unoptimized runs in < 1 second on the VM (hosted on an 8-year old iMac) with the largest input.

If your solution is running too slow, consider what sort of optimizations you can make to save time. Think about your program: in large input cases what steps are taking the longest? How can you improve them?

Sample Input 1

```
18
ls f1
cd dir1
ls f2
cd subdir2
ls f3
cd ..
ls testfile
ls f2
ls f3
cd subdir1
ls f2
cd subsubdir1
cd ..
cd ..
cd ..
ls myfile
cd dir2
cd subdir3
```

Sample Output 1

```
/ f1 myfile
/dir1 f2 f3 testfile
/dir1/subdir1 f2
/dir1/subdir1/subsubdir1
/dir1/subdir2 f3
/dir2
/dir2/subdir3
```

Observe: Even though the directories `/dir1/subdir1/subsubdir1` and `/dir2` contained no files, we still report them because we saw them when analyzing the bash history file.

Sample Input 2

```
13
cd dir1
ls f3
ls f1
cd ..
cd dir2
ls f1
ls f2
cd ..
cd dir1
cd newdir
ls filename
cd newsubdir
ls newfile
```

Sample Output 2

```
/
/dir1 f1 f3
/dir1/newdir filename
/dir1/newdir/newsubdir newfile
/dir2 f1 f2
```

Observe: Even though there was no file directly in the root directory, you still report it in the output. The root directory `/` is **always** reported.

Grading Comments
Despite the fact this appears similar to a morning problem, it will be graded like a weekly exercise. In particular:

- Style matters. Use appropriate comments, proper indentation, etc. Include a file header. Consult the style guide on eClass.

- You must adhere exactly to the output specification: for example, if you output the lines in the wrong order or print extra whitespaces then you will receive a deduction. For full credit, the test centre must accept the output without any presentation error.
- You were only give a few test cases in the test centre files on eClass. We will test your solution on additional test cases that adhere to the input specification.
- Partial credit may be obtained if your solution works on some inputs (eg. small instances).
- Adhere closely to the submission instructions for the weekly exercise. See the eClass code submission link for details.