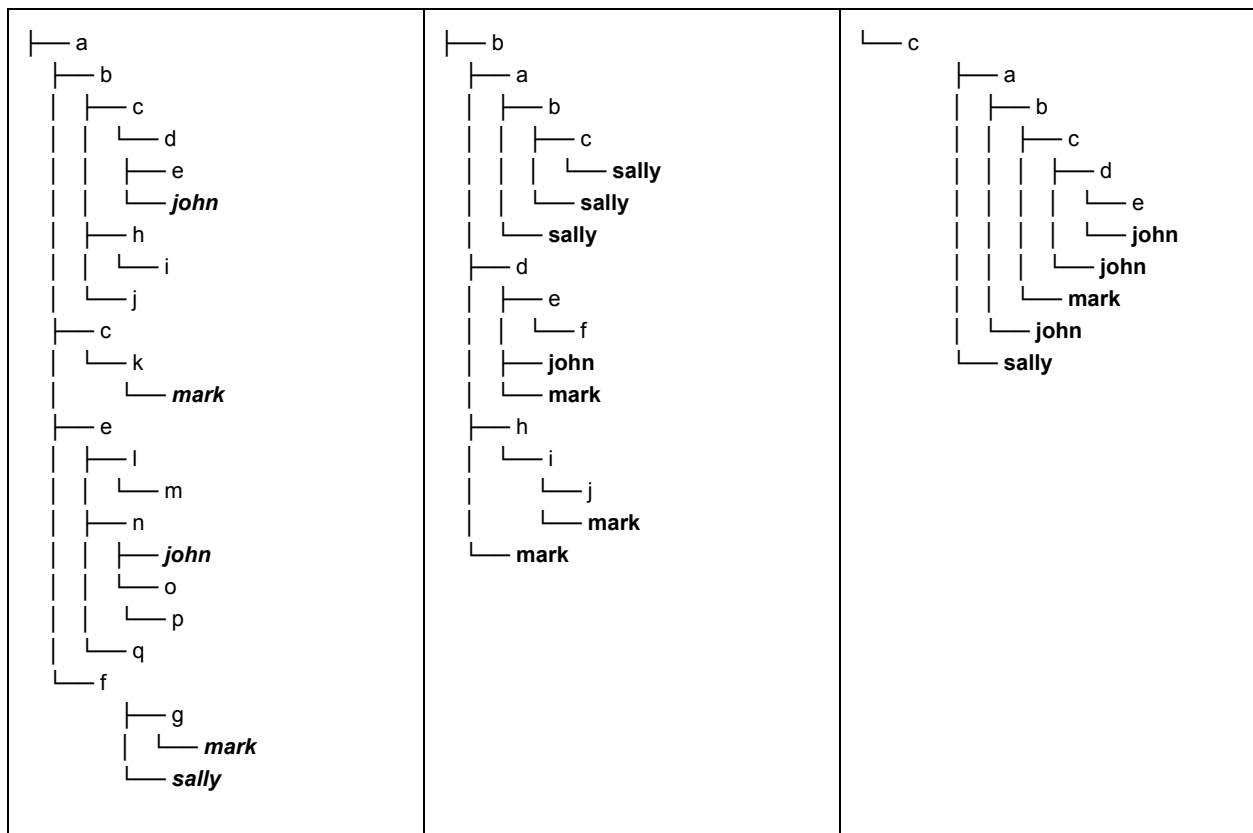## Concurrency and Data Structures

Given a file structure comprised of a finite set of member files, write a multithreaded program such that each thread's job is to traverse the next available top level directory and create an entry in a database each time a file is encountered. The thread should also maintain a count in memory for each time a filename has been inserted into the database.

The thread will then wait until a directory is available (even if that directory has already been traversed) and then repeat the process.

No two threads should attempt to traverse the same directory. The directory structure will not change during test execution.

As an example, configured with a path to an execution root containing the following structure:

```
├── a                          ├── b                          └── c
│   ├── b                       │   ├── a                          ├── a
│   │   ├── c                   │   │   ├── b                       │   ├── b
│   │   │   └── d               │   │   │   ├── c                   │   │   ├── c
│   │   │   ├── e               │   │   │   │   └── sally           │   │   │   ├── d
│   │   │   └── john            │   │   │   └── sally               │   │   │   │   └── e
│   │   ├── h                   │   │   └── sally                   │   │   │   └── john
│   │   │   └── i               │   │   ├── d                       │   │   └── john
│   │   └── j                   │   │   ├── e                       │   │   └── mark
│   ├── c                       │   │   │   └── f                   │   └── john
│   │   └── k                   │   │   ├── john                    └── sally
│   │       └── mark            │   │   └── mark
│   ├── e                       │   ├── h
│   │   ├── l                   │   │   └── i
│   │   │   └── m               │   │       └── j
│   │   ├── n                   │   │           └── mark
│   │   │   ├── john            │   └── mark
│   │   │   └── o
│   │   │   └── p
│   │   └── q
│   └── f
│       ├── g
│       │   └── mark
│       └── sally
```

if your program terminated after each thread completed it's task only once, you would have the following counts stored to memory:

John - 6
Sally - 5
Mark - 6

While your database would contain the following entries:

timestamp	filename	top_level	thread_id
####-##-## ##:##:##	john	a	####
####-##-## ##:##:##	mark	a	####
####-##-## ##:##:##	john	a	####
####-##-## ##:##:##	mark	a	####
####-##-## ##:##:##	sally	a	####
####-##-## ##:##:##	sally	b	####
####-##-## ##:##:##	sally	b	####
####-##-## ##:##:##	sally	b	####
####-##-## ##:##:##	john	b	####
####-##-## ##:##:##	mark	b	####
####-##-## ##:##:##	mark	b	####
####-##-## ##:##:##	mark	b	####
####-##-## ##:##:##	john	c	####
####-##-## ##:##:##	john	c	####
####-##-## ##:##:##	mark	c	####
####-##-## ##:##:##	john	c	####
####-##-## ##:##:##	sally	c	####

A MySQL database with a single table of the structure represented in this example is fine.

Each thread will continue to carry out it's task during program execution. The number of threads and program execution time should be configurable. All threads should be allowed to complete their task upon reaching the specified program execution time, and the counts in memory should be both displayed to the screen and written to a file. The max number of unique filenames in the directory structure will be 10,000.

Use Java, C++, or C# to complete the exercise, and feel free to use any open libraries that you'd like. Include with your submission execution instructions, a sample data set, and expected results. If you have

any questions or clarifications (even over the weekend), please submit to: james@asperasoft.com

We look forward to receiving your solution!