nash TECHNOLOGIES

Imagine you must handle information about *pattern calls*, which later should be displayed to the user. A pattern call is a tuple consisting of a unique integer identifier (“id”), a user defined name (“name”), a project relative path to the so called pattern file (“patternFile”) and a convenience flag, which states whether the pattern should be called or not (“called”). An example tuple is as follows:

42, “myPattern”, *src/patterns/Functional.pat*, false

Write a solution in C++ which holds pattern call tuples in memory, is able to store a set of tuples to a file (resp. read a set from a file), and answers the following queries:

- retrieve a pattern call with a specified identifier, e.g. 42
- list all pattern calls with a specified name, e.g. “myPattern”
- list all pattern calls with a specified path, e.g. “src/patterns/Functional.pat”
- list all pattern calls which **are** skipped, i. e. when the “called” flag is false
- list all pattern calls which **are not** skipped, i. e. when the “called ” flag is true

The number of tuples are not known beforehand, but are assumed to fit in-memory. In addition, the tuples will not be modified after initialization. The code you deliver should be **production ready**, i.e. maintainable and documented.

Finally, write a simple [C++] program that demonstrates the implemented interfaces. It should create a few tuples with dummy data, store them in a file, read the tuples back from the file and check that the above mentioned query interfaces work correctly.

Questions to be answered:

- *What are the benefits of your design?*
- *Do you see improvement potential?*
- *What assumptions did you make and what trade-offs did you consider?*
- ***What is the complexity (Big-O notation) of the queries you provide?***
- *Which part of your solution took the most time (e.g. design, coding, documentation) and why?*

Please send us your complete solution in a zip file and the answers to the questions above. If you have any further questions please let us know. Otherwise, we look forward to seeing your solution!