Core Services Engineer tests

Below test will be given to candidates applying for engineering positions particularly focusing on back end/server-side development. We intend to provide a focused test that the candidate can complete during a weekend or half-day at our offices. The focus is code quality, design approach ( data structures, re-use of code/recursive code, cyclomatic complexity) and problem-solving skills.

The candidate must complete the sections under “Must do”. Also, they can try out the “additional scope” as well if they wish to exhibit additional expertise.

# **Problem statement**

**Must dos:**

Model a family tree in a tree like data structure. The candidate can select an appropriate physical data structure to store the family tree but must explain why a particular data structure was selected. The logical structure of each family member node can be like:

*{ name, age, parents ( reference to maximum two parent node), kids ( reference to many kid nodes)}*

Note: The candidate must try to implement the data structures using first principles and avoid using the high-level data structures provided in Java (e.g. Sorted lists).

The candidate must try to implement below solutions with algorithms of O(n) time complexity (linear )

**Q1:** Implement a test class to inject and create the family structure one node at a time. The structure should at least have five levels ( great grant parents, grand parent, parent, and kids, grand-kids).

* Make the tree unbalanced. Try to have a good distribution of the number of kids for a pair of parents (i.e. some with no kids, some with 2 kids, some with 3, or 4….).

The injection time complexity should be O(n). i.e. one logical search to find the place to enter a family node.

**Q2:** Implement an algorithm with possibly O(n) complexity to sort the whole family tree in age descending order.

**Q3:** Implement a similar algorithm to sort the whole family tree in age ascending order.

**Q4:** Pretty print algorithm to print the family tree.

**Q5:** Print the reverse family tree ( upwards) from a node including both parents for each level(i.e. for a kid print its parents, parents’ parents, great grand-parents.

**Q6:** Can you think of an algorithm that can insert a new family member (assume we found a long missing member!) in to the correct place of a sorted (ascending) list based on age, where the algorithm’s time complexity is better than O(n)?. pseudo code or actual implementation is fine.

**Q7:** Candidate can use Mockito like mocking tool to showcase how the data persistence layer ( assume these records can be persisted in a DB but for the test we don’t need a working database) can be mocked.

Ideally candidate can use Maven or Gradle as the build tool.

**Languages:**

* Java, Junit, Mockito

**Additional scope:**

The candidate can perform this additional scope to

1. Wrap the above implementation in a **Spring boot app exposing RESTful services** to
   1. Enter individual family nodes (HTTP POST)
   2. To get sorted family list (HTTP GET) for ascending and descending
2. The candidate can use Swagger like API documentation specification to show the RESTful service definitions.
3. Implement a good optimistic concurrency model to the data structure to avoid parallel updates to the same sub-tree of a family tree. (i.e. modifying kids of a pair of parents).

# **What needs to be submitted**

1. Working code
2. Instructions on how to run and test the code
3. Answers for the design question like why a particular data structure was selected.

# **Sharing the work**

The candidate can zip the project folder and email to us. If not, they can commit the code to a git repository and share the details for us to download.

# **Feedback and verification**

1. We will evaluate the answers provided for the questions, evaluate the design and run the code.
2. We will also verify the authenticity of the code.