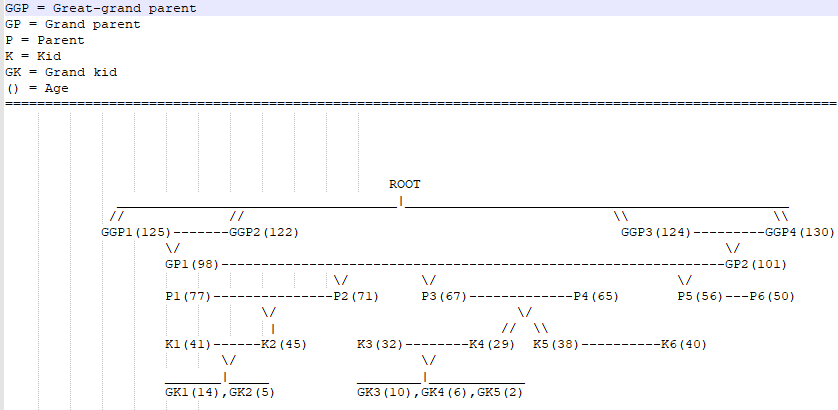
Please refer to the code at the git repository: <https://github.com/c7chris/familytree/tree/feature/initial>

The application can be run as a regular Spring Boot application using:

mvn spring-boot:run

Alternatively, it can be run as a Java application using the main() method in com.chris.familytree.MyFamilyTree, which includes some test data.

A default root node is assigned to all those family members who do not have parents. For instance, a sample structure of the family tree would be as follows. This sample has been used in the tests.



The family tree nodes are represented by Java objects, in org.chris.datastructures.FamilyTree and links to other nodes (like parents and kids) are through references stored in the object. The set of family member that have spouses are noted in the HashMap partnerMap.

**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.

Please note the test class com.chris.familytree.FamilytreeApplicationTests creates such a family tree. It is at the most *O(n)* because at the most, a search would be done for the parent of node to be inserted. If the node has no parent, no search would be done since it would become a child of the root node.

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

Please refer to org.chris.datastructures.FamilyTree.sort(String sortOrder). This method calls another overloaded method which does the sorting recursively in at most *O(n)*, because if the current node has a partner, the partner is fetched from the partnerMap in *O(1)*.

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

(Same as the previous answer) Please refer to org.chris.datastructures.FamilyTree.sort(String sortOrder). This method calls another overloaded method which does the sorting recursively in at most *O(n)*, because if the current node has a partner, the partner is fetched from the partnerMap in *O(1)*.

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

Though I could not complete it, the idea is to get the maximum depth and breadth of the family tree and use them when indenting the family member whilst printing. I have part of the code for finding the depth: org.chris.datastructures.FamilyTree.determineDepthOfFamilyTree()

**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.

Please refer to org.chris.datastructures.FamilyTree.printChildAndAncestors() which first prints the child node and then calls printAncestors() to print the ancestors recursively.

**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.

We could use a binary search recursively: that is, start in the middle of the list and then move left or right depending on whether the new family member’s age is less than or greater than the middle element.

Method insert(newFamilyMember, start, end, list) {

If(start – end = 0) {

Add family member in current ‘start’ position.

Else If (newFamilyMember.age < middleElement.age) {

insert(newFamilyMember, 0, position(middleElement) -1, list)

}

Else insert(newFamilyMember, position(middleElement), list.size, list)

}

**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.

Please refer to com.chris.familytree.service.impl.FamilyTreeServiceImplTest which tests the service class by mocking the data persistence layer.

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

Please refer to com.chris.familytree.controllers.impl.FamilyTreeControllerImpl. For some sample requests please import the below file into Postman and run.



1. The candidate can use Swagger like API documentation specification to show the RESTful service definitions.

When run as a Spring Boot application, the Swagger will be available at: <http://localhost:8080/swagger-ui.html>

1. 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).

We could allow only one update at a time to take place in the sub-structure, by making sure that for a given parent, only one update goes through. We can allow multiple reads though. We also need to make sure that only the immediate parent is the one that controls modification to its immediate children, to prevent unnecessary locking out of a larger part of the family tree – in other words, there would be no need to restrict access from the grand-parent or great-grand-parent level, since this might lock out other children who are not currently being modified.