Requirements for Genealogy App

1. Summary
   1. The genealogy app begins by reading a provided genealogy document, and then creating a family tree based on this data. Once the family tree has been created, it should provide a number of quality of life features to help the user read and navigate the tree, such as the ability to search for members of the tree or add new members.
   2. We want to create an app that clearly and easily keeps track of relationships between people and family genealogies or family trees, providing ancestors and descendants of the person searched for.
2. Scope/Demographic
   1. As far as scope, we are trying to be conscious of requirements creep and not let it get too out of hand. Our three main functions are the read in file, the search function, and the addition of new family members to the database.
3. Searching
   1. The user should be able to search up a full name, first name or surname and the program will tell them whether that name is present in the database or not.
   2. If a user searches up a surname or full name contained in the database, the app should show them a list of individuals with that full name or surname. The user can also search for any attribute other attribute (birthplace, marriage status, children, etc.) and have the same results.
4. Viewing the Family Tree
   1. Since children result from partnering between the child’s biological parents, the app includes records of partnership and any offspring associated with that partnering. Children are related to both their biological parents and parents may be in more than one partnership at the same or different times. It is required to trace all children of each person across all partnerships. The user should be able to access any relationship via one person.
   2. It is likely that some members of the tree will be unknown so there will be placeholders for them.
5. Design
   1. The program holds 5 modules within the src package:
      1. GenealogyApp
         1. Main code, contains GeneDataBase, OutputFile, and User objects.
      2. GeneDataBase
         1. Processes file at read in, generates HashMap<String id, Person p> data structure for family tree, creates Person objects with all known data stored, and appends relationships to Person objects.
      3. OutputFile
         1. Copies HashMap that is stored in GeneDataBase and writes all data to a temporary file that the user may save when they are finished using the app.
      4. Person
         1. Contains attributes for all known data about an individual and the relationships the individual shares with other Person objects in the data set. Additionally, calculates the age of a person, deceased or not.
      5. User
         1. User GUI that will allow data entry through: checkboxes, drop down selection bars, click-commands, or keyboard entries.
   2. When the user begins the app, GenealogyApp will create three objects: GeneDataBase gdb, OutputFile op, and UserGUI user. GenealogyApp will contain the main to better control interactions between classes. GeneDataBase will read in “FamilyTreeInputTextV2.txt” and generate a HashMap<String, Person> with all people (Person class) made. GeneDataBase will continue processing the file to create relationships for the individuals in the map. After, the user may add people, search the tree, or make changes presented through a number of components in the UserGUI. Once the user has indicated they are finished, an output file of the final HashMap, with user additions/interactions, will be made for the user to save if they would like, and the program will close.
6. Database
   1. The database should differentiate between partnerships and children of said partnership.
   2. The database should find different types of relationships between its entries, i.e. aunts and nieces, grandparents and grandchildren, etc.
   3. The database should allow for “gaps” in information, such as unknown parents. In this case, we should still be able to relate siblings.
   4. The user should be able to add people to the database.
   5. For each person entry in the database, certain attributes are stored if applicable:
      1. ID
      2. Given name
      3. Family name
      4. Date of birth
      5. Date of death
      6. Birthplace
      7. Death Place
      8. Age
      9. Suffix
      10. Labels defining if they are a child of a partnership
   6. For each partnership entry in the database, certain attributes are stored if applicable:
      1. The children of the partnership
      2. Marriage start date
      3. Marriage end date
      4. Marriage location
7. Use Cases
   1. We have been given a file with family tree information. The program should read the information, interpret what kind of information it is reading (i.e. birthday, name, etc), store the information in code that is easily accessible, and then organize the information into a family tree format easily readable to the user.
8. Functional Requirements //pre/post conditions, more technical
9. Determine if a given person is known to the app. Provide at least lookup by name (names do not need to be unique as they often repeat in families across generations). Likely useful to be able to search by family name, given names, date ranges, etc.
10. Add a person to the app. It may be necessary to connect them to their parents or children if these are already known in the app(update or add partnering).
11. Read in a text file of current people, their family relationships, and their partnering history. Upon request save any changes made back to a text file. We’ll provide you an example text file to work with and the format.
12. Record the start and end of a partnering relationship.
13. Record a new child(ren)of an existing or new partnering.
14. For any person, find their parents, grandparents, etc. (for all those known by the app).
15. For any person, find their children, grandchildren, etc.
16. For any person, find their first cousins(first cousins have the same grandparents).
17. For any person, find their current partner(s).
18. For two people determine if they are related. Two people are related if they have a common ancestor somewhere “up” their family trees.