Harrisburg University of Science & Technology

CISC 610 Data Structures & Algorithms

Instructor: Caleb Druckemiller, M.S.

Assignment 5: Graphs

Family tree's and genealogy software has become more and more prevalent in recent years. From the name you might expect that a family tree would be easily represented by a tree structure, but that is not the case! A more appropriate data structure to represent a family tree would be a type of graph. Using the description of the family that accompanies this assignment, you must represent this family using a graph structure. The graph needs to be a weighted graph. The weights will constitute the types of relationships, I recommend using some kind mapping between numbers and strings to represent the relationships. When adding family members to the graph, this can be done programmatically for the provided family members within the description file.
Additionally, I also want there to be an interface in which a user can create a new family member and add them to the tree. This can be a simple CLI where the user provides a name, gender, and age to create a person. Then another simple CLI where they select which member of the family they want the original relationship to be with and what kind of relationship it should be. Finally, they can edit the family member using another CLI and selecting the family member they wish to edit, the operation they wish to perform (edit name, edit age, edit relationship), and then add new relationship between family members which can call a function that you create in order to add the original relationship. Remember the DRY philosophy, where code can be modularized or made into a function, it should be if you plan on using the logic again.
Finally, I want you to make data assertions within the `FamilyTree` class that enforce certain "rules" that exist in a typical human family. An example would be a person should not have any kind of relationship to itself (a person can not marry themselves, a person can not be their own brother, sister, father, mother, etc.). There should be at least 3 data assertions. These should exists as part of the family tree, not as part of the graph.

As a hint, for a successful design: I would recommend using layers of abstraction. Your graph class is the backing structure to the family tree class. Your family tree should implement methods that interface with the graph class, i.e. `add_family_member()` should call the constructor to create a node and then call a function within the graph class to add a node to the graph. Then using the relationships function parameter, you can add edges to the graph between the new nodes and the existing nodes. The family tree should be what enforces what relationships can exist through the data assertions, the graph does not care about what relationships are made between family members. Your functions that the *user* would interface with would be greatly reduced compared to the total number of methods within the classes themselves. The user should be able to add, remove, and modify family members and that's about it. Therefore those should be your function calls.

Submission Goals

- (120 pts.) Create a `FamilyTree` class that will represent a family tree for a given family.
  - The class should contain several types of relationships that commonly happen within a family (siblings, marriage, offspring, etc.)
- (40 pts.) Programmatically add the family members to the graph as described by the accompanying family description file.
- (40 pts.) Give data assertions to the `FamilyTree` class to enforce restrictions for basic family structure (at least 3); i.e A person can not marry themselves.
- (40 pts.) Provide a simple CLI the enables users to add, remove, and edit family members.

* Note that there is a total of 240 points on the table here. The assignment is only worth 200 points. You have 40 points of optional goals.

Your submission should be accompanied by a 8 minute walk-through of your code. This analysis should include your decision making process, the logic behind you code, an your original thoughts that went into the decision making on why your code is written and performs in the manner in which you have written it. If you can not adequately explain how your code functions, it is difficult to believe that you created it yourself as it is inherently difficult to make that which you don't understand.
All video submissions must:

- Be narrated by your own voice - Silent submissions will not be considered
  - If you need accommodations regarding your voice recording, reach out to me **BEFORE** the due date of the assignment
- Capture your screen to include the source code and other assets required by the assignment if necessary for the comprehension of your explanation.

Your submission should include:

- A link to your YouTube/Loom Video upload submitted as a `.txt` file or as a submission comment **OR** a video file (`.mp4` preferred).
  AND
- Your code project (source code, resource files, etc.) unzipped.
- If necessary, provide a README file if an explanation is required to execute your code.