Lab Class 1: C++ Primer

Instructor: Dr. Rupesh Nasre.

The goal of this class is to give a brief introduction to C++ and get you started for the lab. The lab lecture will cover the following topics:

- C++ basic syntax.
- Classes, objects, fields, methods.
- Terminal and file input/output.
- Singly linked list insert, delete, search.

Assignment: We are interested in maintaining a scientific genealogy of the scientists according to their supervision relationship. In particular, we are given the information about a scientist and his academic advisor. However, to simplify matters, we will assume that any scientist has supervised at most 2 advisees (students). Note that the advisees of the scientists may themselves be supervisors for other advisees.

Use a binary tree to represent this academic genealogy.

Your program should accept 2 input files – a) input file, b) test file. The input file contains a list records each consisting of name of scientist (UNAME), the user ID (UID) along with his/her Parent ID (PID). Assume that the input-file contains lines of the format:

PID UID UNAME
PID UID UNAME
....

Assume that UID and PID are integers and the PID of the root is 0 and UNAME is a string without any spaces. Your program should be able take the filenames as an input arguments and create a binary tree containing all the valid records in the input file. The data structure (binary tree) should support the following operations.

- **AddAdvisee (PID, AdviseeID, Name):**
  Add an advisee with id AdviseeID and name Name as an advisee of Scientist with ID as PID. Make sure that the procedure can handle cases like:
  - Adding an advisee to a PID which is not present.

- Adding an advisee to a PID which already has 2 advisees.
- Adding a Scientist as a root (with PID = 0) when such a root already exists.

In all the above cases the program should flag an appropriate error. Such inputs are erroneous records all others are valid records.

- **ListAllAdvisees (UID):**
  Print ID and NAME of all the people who are directly or indirectly adviseed by the scientist with ID as UID. Flag an error if UID does not exist.

The test file will contain records of 2 types:

Add UID PID NAME
List UID

The "Add" kind of record should have the above mentioned format and should invoke a AddAdvisee(PID, UID, NAME). Similarly "List" type of record should invoke a ListAllAdvisees(UID). For any other record type, your program should give an error message and proceed with processing of the rest of the records.