BLG 233E, Data Structures and Laboratory, Fall 2012
Assignment #3

Asst. Prof. Dr. Gülşen Cebiroğlu Eryiğit
Res. Asst. Doğan Altan

Due: Tuesday, December 25 , 23:00 [Hard deadline. Ninova HW submission closes automatically. Late homeworks will NOT be accepted.]

Required submission components: **1)** your source code file (.cpp) and **2)** a soft copy of your report (in pdf format).

NINOVA SUBMISSION: You should be aware that the Ninova system clock may not be synchronized with your computer, watch, or cell phone. Do not e-mail the teaching assistant or the instructors with your submission after the Ninova site submission has closed. If you have submitted to Ninova once and want to make any changes to your report, you should do so before the Ninova submission system closes. Your changes will not be accepted by e-mail. Connectivity problems to the Internet or to Ninova in the last few minutes are not valid excuses for being unable to submit. You should not risk leaving your submission to the last few minutes.

CHEATING: This is not a group assignment. It should be done individually. When a student receives information from another person about a program, it is considered cheating when the information is enough to precisely describe the code in a nontrivial part of the program. For the most part, oral discussions between students about the algorithms used in a program or the program's design, is not considered cheating. The most common example of cheating occurs when a student copies all or part of a program from another student and then changes the names of some of the program variables and functions. If cheating is discovered, a report will be made recommending a course grade of "VF".

HOMEWORK DESCRIPTION

In this homework, a family will be represented using a tree. Each individual will be represented as a node in the tree, and relationships will be represented with the connections between the nodes. There are two types of people, namely, individual and spouse. Their member variables are described in more detail below.

- Each individual will have a **name** (char*), **year of birth** (int), **gender** (char), **pointer array to children** (node**) (you may assume that a person can have at most 5 children), **pointer to wife/husband** (spouse*), **node ID** (int), and **parent ID** (int).
- The spouse type will have two data types: **name** (char*) and **year of birth** (int). Do not add any extra variables to the given structs. Note that in this family tree, we ignore the families of the spouses (i.e., their sisters, brothers, and parents). We do not include their descents in the tree.

1. Input File Properties

You will create the tree using the given input file, “family.txt”. For simplicity, each person’s parent ID is given in the file so that you can find the related node and add the node to the tree. **However, you should not use the parentIDs or the nodeIDs of the nodes after creation of the tree (You should use these values only for creating the tree; you should not use them for further steps of the homework).**

- Representation of **individual** type in the file

Deniz 1940 Male 3 0 -> Name, year of birth, gender, nodeID, parentID

- Representation of **spouse** type in the file

spouse Tacettin 1 1933 -> special key word for defining that a person is spouse, name, spouse’s nodeID, year of birth

2. Discovering the Relationships

In this part of the homework, you should write some functions to discover the relationships in the family.

- **printParent(parameterName):** This function should print the names of the parents (both mother and father) for the given parameter. If the parameter has no parent, then the necessary message should indicate this to the user.

- **printChildren(parameterName):** This function should print the names of the children for the given parameter. If the parameter has no children, then the necessary message should indicate this to the user.

- **printCousins(parameterName):** This function should print the names of the cousins for the given parameter. If the parameter has no cousins, then the necessary message should indicate this to the user. Here, cousins are only the people who have the same grandparents.

- **printGrandchildren(parameterName):** This function should print the names of the grandchildren for the given parameter**.** If the parameter has no grandchildren, then the necessary message should indicate this to the user.
- **printGrandparents(parameterName):** This function should print the names of the grandparents for the given parameter**.** If the parameter has no grandparents, then the necessary message should indicate this to the user.

- **printSiblings(parameterName1**): This function should print the names of the siblings for the given parameter**.** If the parameter has no siblings, then the necessary message should indicate this to the user.

- **isSibling(parameterName1, parameterName2):** This function returns true if the two parameters have the same parents.

3. Printing the Family to the Screen
In this part, you will print the members of the family to the screen. You should print the family tree layer-by-layer.