CMPE 221_223_242
Spring 2023
Programming Homework 3

This assignment is due by 23:59 on Sunday, 28 May 2023.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:

   - 18.05.2023, 18:00- 19:00, https://tedu.zoom.us/j/97572713121
   - 25.05.20023, 18:00-19:00, https://tedu.zoom.us/j/94351670462

Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK

Q1(50 points):

In this question, you are expected to implement a simple family tree system **using tree data structure**. You must use your own implementation of tree data structure by taking inspiration from your textbook or web. You are not allowed to use any external library or .jar file.

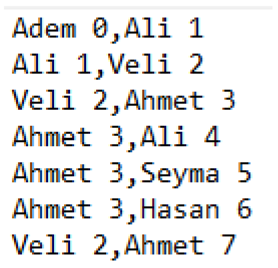
In your implementation, you should keep name and ID for each member.

You should implement following operations:

1. Create family tree
2. Print All Descendants
3. Check Ancestor
4. Check descendant
5. Check siblings
6. Find oldest common relative
7. Quit

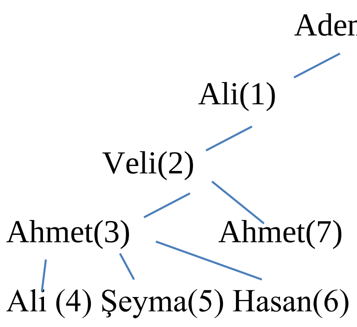
Create family: In this function, first you should read txt filename which contains persons. Then, you should create an appropriate tree for given txt file. You should call this function at the beginning of the main.

Each line of txt files contains two person names with their ids. ID is the unique identifier.



```
Adem 0,Ali 1
Ali 1,Veli 2
Veli 2,Ahmet 3
Ahmet 3,Ali 4
Ahmet 3,Seyma 5
Ahmet 3,Hasan 6
Veli 2,Ahmet 7
```

Above data should generate the following family tree:



Print All Descendants: This function, takes integer ID as an input and prints all descendants in the ascending order based on ID.

PrintAllDescendants(1):

Veli, Ahmet, Ali, Şeyma, Hasan, Ahmet

Check Ancestor: This function takes two ID arguments as inputs. You should print true if the first ID is ancestor of the second ID.

CheckAncestor(1,2) -> True

CheckAncector(2,1) -> False

CheckAncestor(5,6) -> False

Check Descendant: This function takes two ID arguments as inputs. You should print true if the first ID is descendant of the second ID.

CheckDescendant(1,2) -> False

CheckDescendant (2,1) -> True

CheckDescendant (5,6) -> False

Check Sibling: This function takes two ID arguments as inputs. You should print true if the first ID is siblings of the second ID.

CheckSibling(1,2) -> False

CheckSibling (2,1) -> False

CheckSibling (5,6) -> True

Find First Oldest Common Relative: This function takes two ID arguments as inputs and prints name of first oldest common relative. The output of this function can not equal to arguments.

OldestCommonRelative(3,7) -> Veli

OldestCommonRelative (4,7) -> Veli

OldestCommonRelative (4,5) -> Ahmet

OldestCommonRelative (1,2) -> Adem

Quit: This function must stop your program.

Example Input/Output:

Below is an example of input/output. Do not forget to test your program with different test codes as well. You can check VPL in the LMS page for more examples.

```
Enter filename:
family1.txt                          // after reading filename, you should call create family
Enter operation code:              //1-> print All descendants, 2->Check ancestor, 3-> Check descendant
                                   //4-> Check siblings, 5-> Find oldest common relative, 6-> Quit
1
Enter ID:
3
Ali, Şeyma, Hasan
Enter operation code:
2
Enter IDs:
1 2
True
Enter operation code:
2
Enter IDs:
2 1
False
Enter operation code:
2
Enter IDS:
5 6
False
Enter operation code:
3
Enter IDs:
2 1
True
Enter operation code:
4
Enter IDs:
2 1
False
```

```
Enter operation code:
4
Enter IDs:
5 6
True
Enter operation code:
5
Enter IDs:
3 7
Veli
Enter operation code:
5
Enter IDs:
4 7
Veli
Enter operation code:
5
Enter IDs:
1 2
Adem
Enter operation code:
6
Stopped!
```

Q2(50 points):

In this question, you should develop a program that manages a company's employee database. Each employee is identified by a unique ID number, name and gender. The database needs to support the following operations efficiently:

- Insert an employee into the database
- Delete an employee from the database
- Search for an employee by ID number
- List all employees in order by ID number
- List all employees in order by ID number in given range.
- Quit

You should use binary search tree data structure to manage the employee database. Your implementation should support the above operations and be as efficient as possible. You would need to think about how to efficiently perform each operation.

InsertEmploye: This function takes integer ID, string Name and Boolean Gender, (0 represents male, 1 represents female) as arguments. You should add a given employe to your database.

DeleteEmploye: This function takes integer ID as an argument and delete it. If the given ID does not exist in your database, do nothing.

SearchEmploye: This functions takes integer ID as an argument and prints all information. If the given ID does not exsist in your database, print "record not found".

ListAllEmploye: This functions takes no argument and prints all information in ascending order based on ID.

ListAllEmployeWithRange: This functions takes two integer arguments which represents minimum and maximum bounds for IDs, respectively. You should print, all employees information which have ID within the specified range in ascending order based on ID.

Quit: This function must stop your program.

Example Input/Output:

Below is an example of input/output. Do not forget to test your program with different test codes as well. You can check VPL in the LMS page for more examples.

```
Enter operation code:             //1-> Insert, 2-> Delete, 3-> search, 4-> List, 5->List with Range, 6-> Quit
1
Enter information:
1234 Bedrettin Male               // Add this record to your database
Enter operation code:
1
Enter information:
12345 John Male                  // Add this record to your database
Enter operation code:
1
Enter information:
12 Khaleesi Female                // Add this record to your database
Enter operation code:
4
12 Khaleesi Female
1234 Bedrettin Male
12345 John Male
Enter operation code:
5
Enter bounds of range:
10 12
12 Khaleesi Female
Enter operation code:
5
Enter bounds of range:
10 11
No record found.
Enter operation code:
2
```

```
Enter ID to be deleted:
12
Enter operation code:
3
Enter ID to be searched:
12
No record found.
Enter operation code:
6
Stopped!
```

WHAT TO HAND IN

A zip file for both parts containing:

- The Java sources for your program.
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You should test your Java source files on (if) available Moodle VPL environment to ensure your code solution's correctness before submitting. VPL simply tests your program's output by checking against given sample input. You should pass that task's VPL test case successfully.
- A **maximum-3 pages** PDF report document that explains your own answers for programming task in a clearly readable PA report format (refer to **PA REPORT FORMAT** section). Include this report in your zip file and upload it to the PA Report Submission screen in LMS.
- For given task, only code or report submission will not be graded. In other words, you should submit both correct code solution and its related report for the task in order to be graded.

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be hand-written. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%2.5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%15): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation, Functionality (%20): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the pseudocode algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section.

Testing (%7.5): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%5): In this final section, you should briefly answer the following questions:

- What were the trouble points in completing this assignment?
- Which parts were the most challenging for you?
- What did you like about the assignment? What did you learn from it?

GRADING:

- Codes (%50, Q1:25, Q2:25)
    - Available test cases evaluation on VPL: %15, (Q1:7.5, Q2:7.5)
    - Hidden test cases evaluation: %15, (Q1:7.5, Q2:7.5)
    - Approach to the problem: %20, (Q1:10, Q2:10)
- Report (%50)
    - Information: %2.5
    - Problem Statement and Code design: %15
    - Implementation, Functionality: %20
    - Testing: %7.5
    - Final Assessments: %5

Submitting report without codes will not be evaluated!!

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on 28.05.2023.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload files and any additional files if you wrote additional classes in your solution as a single archive file (e.g., zip, rar).

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-----------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-----------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
void setVariable(char varName, int varValue)
//-----------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
// Postcondition: The value of the variable is set.
//----------------------------------------------------------
{
      // Body of the function
}
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TA, Bedrettin Çetinkaya. Thus, you may ask her your homework related questions through <u>HW forum on Moodle course page</u>.