CMPE 343
Fall 2023
Programming Homework 4

This assignment is due by 7 January 2024, 23:59.

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:

    - Recitation 1: 18.12.2023, 19:00-20:00 https://tedu.zoom.us/j/98990267511
    - Recitation 2: 25.12.2023, 19:00-20:00 https://tedu.zoom.us/j/98990267511
    - Recitation 3: 02.01.2024, 19:00-20:00 https://tedu.zoom.us/j/99642469651

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

In this homework, you must implement your own graph data structure by taking inspiration from your textbook and use it to help to solve problem. You are not allowed to use any external library or .jar file. Any solutions without using trie data structure are not evaluated!

In this homework, you are expected to implement Trie data structure and following functions. Firstly, you should create a Trie data structure, and then you should insert all words read from the user. You can implement more than one Trie data structure. Finally, you should complete following functions:

(10 points) Void Search (String arg): This function should print true if given argument is in your Trie, otherwise it should print false.

(20 points) Void countPrefix(): This function takes all the strings in Trie and checks if a string is a prefix of other strings in Trie. It prints the number of occurrences for each string. For this function, you may use a symbol table that keeps track of the number of appearances of each key. Also, you should print prefix in alphabetical order of corresponding string.

Note: You can change the parameter for this method if you need!

(20 points) Void reverseFind (String suffix): This function should print all strings end with given suffix in your Trie, lexicographically. For this function, you may consider using multi-trie solution, or research and use more complex data structures such as suffix arrays. Also, you should print string in alphabetical order.

(25 points) Void ShortestUniquePrefix (Trie trie): This function should print shortest unique prefix to identify each string in your trie. If there is not any unique prefix, print “not exists” .

Note: You can change the parameter for this method if you need!

(25 points) Void LongestCommonPrefix (): This function should print longest common prefix for all strings in your trie. If there is not any unique prefix, print “not exists” .

Note: You can change the parameter for this method if you need!

Sample Input 1:

```
7                    // number of inputs
at                   // word 1
atomy                //word 2
ate
ato
borned
born
curve                // word 7
Enter Operation Code:
//1- Search, 2- countPrefix,
//3-  reverseFind, 4- ShortestUniquePrefix
// 5- LongestCommonPrefix, 6- exit
1                    //call Search Function
ate
True
Enter Operation Code:
1
Atel
False
Enter Operation Code:
2                    //call countPrefix
at: 3
ate: 0
ato: 0
atomy: 0
born: 1
borned: 0
curve: 0
```

```
Enter Operation Code:
3                    // call reverseFind
e
ate
curve
Enter Operation Code:
3
ned
borned
Enter Operation Code:
4                    // call ShortestUniquePrefix
at: not exists
ate: ate
ato: ato
borned: borne
born: not exists
curve: c
Enter Operation Code:
5                    // call LongestCommonPrefix
not exists
Enter Operation Code:
6
```

Sample input 2:

```
3                   // number of inputs
at                  // word 1
atomy               //word 2
ate                 //word 3
ato                 //word 4
Enter Operation Code:
//1- Search, 2- countPrefix,
//3-  reverseFind, 4- ShortestUniquePrefix
// 5- LongestCommonPrefix, 6- exit
5                   //call LongestCommonPrefix
at
Enter Operation Code:
6
```

You can find example inputs/outputs on VPL.

WHAT TO HAND IN

You should submit your codes to VPL and your report (in pdf format) to submission area.

- 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,)
    - Available test cases evaluation on VPL: %15,
    - Hidden test cases evaluation: %15,
    - Approach to the problem: %20,
- 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 07.01.2024 - 23:59 .

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload your codes to VPL and your report to Submission area in .pdf format.

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 HW forum on Moodle course page.