CMPE 343
Spring 2023
Programming Homework 4

This assignment is due by 23:59 on June 7, April 2023.

You are welcome to ask your HW related questions by joining the recitation hours given below:

There will be two Q&A Office Hours on the following days:

- CMPE343-HW4-OfficeHour1: May 31, 06:00-07:00 PM: https://meet.google.com/vtk-bouq-nyu
- CMPE343-HW4-OfficeHour2: June 2, 06:00-07:00 PM: https://meet.google.com/jrr-vyta-tjq

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 part, 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 graph 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 user. You can implement more than one Trie data structure. Finally, you should complete following functions:

Question 1(25 points):

Boolean Search (String arg): This function should return true if given argument is in your Trie, otherwise it should return false.

Void countPrefix(Trie trie): 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.

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

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.

Sample Input:

```
7                   #number of inputs
br                  #word 1
bridge
brick               …
dogma
cat
cathedral
dog                 #word 7
```

Sample Output-1

```
Choose the function you want to use:

1) Search
2) Count Prefix
3) Find Reverse

1
cat

True
```

Sample Output-2:

```
Choose the function you want to use:

1) Search
2) Count Prefix
3) Find Reverse

1
ball

False
```

Sample Output-3:

```
Choose the function you want to use:

1) Search
2) Count Prefix
3) Find Reverse

2

2 0 0 0 1 0 1
```

Sample Output-4:

```
Choose the function you want to use:

1) Search
2) Count Prefix
3) Find Reverse

3
e

bridge
```

Sample Output-5

```
Choose the function you want to use:

1) Search
2) Count Prefix
3) Find Reverse

3
ab

No result
```

Question 2(25 points):

In this question, you will implement a String sorting algorithm. You will be given two same length of arrays of String. The Strings of the second array will not contain duplicate characters. You will sort the first array based on the following rules.

First, you need to find the distance between the Strings at the same index of the arrays. The distance calculation formula is:

- Each letter has a number starting from 1 and increasing one by one. "A" is 1, while "Z" is 26.
- The integer value of a String is determined by the letter number. For an example, the integer value of "abj" is 1210.
- Find the distance between two Strings at the same indexes by subtracting their values. For example, the distance between at index "0" from the first array "abj" and from the second array "bal" is |1210-2112| = 902.
- If the distance value is even, then sort your String from the first set based on the String from the second set. In our example "abj" will be sorted based on the order of "bal", which 'b' has the highest priority while "l" has the least in sorting. The sorted version of "abj" will become "baj". The order of letter "j" in the first string cannot be found in the second string. Therefore, you need to add it to the last of the String.
- If the distance value is odd, then sort your string lexicographically in ascending order.

Example-1

```
First Array:
dog honey apple rope

Second Array:
gdbo bonex pina elo

Sorted Array:
gdo ehnoy ppale eorp
```

Example-2

```
First Array:
forest water nick doze

Second Array:
ftrki olis cim ipgk

Sorted Array:
eforst aertw cink doze
```

WHAT TO HAND IN

- **You need to upload your code into VPL on LMS for each question.** If you do not upload your code into VPL on LMS, your homework will **not be evaluated.**
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You need to upload **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).
- You can work as a group with a maximum of 1 person. It is enough for one of the group members to submit the homework.

PA REPORT FORMAT

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be handwritten. 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 and 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: %25 for Q1 and %25 for Q2)
  - Available test cases evaluation on VPL: %15
  - Hidden test cases evaluation: %15
  - Approach to the problem: %20
- Report (%50: %25 for Q1 and %25 for Q2)
  - Information: %2.5
  - Problem Statement and Code design: %15
  - Implementation, Functionality: %20
  - Testing: %7.5
  - Final Assessments: %5

IMPORTANT

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

1. This assignment is due by 23:59 on Wednesday, June 7th.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload your code files into VPL and your report.

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, Enes Arslan. You are welcome to ask your course instructors Tolga Kurtuluş Çapın and Ulaş Güleç for help as well as TA.