CCDSALG Term 3, AY 2019 – 2020

Project 3 Documentation – Word List (Binary Search Tree Application)

Section	Names	Task 1	Task 2	Task 3	Task 4	Task 5
S12	Gan, John Matthew Ong		X	X	X	X
S16	Noblefranca, Jose Noel Cleofe	X	X	X	X	
S15	Remudaro, Angelo Alvarez	X		X	X	X

Fill this part with your section and names. For the tasks, put an X mark if you have performed the specified task. Please refer to the project specifications for the tasks.

1. Programming Language Used: Java
2. Why did you choose the programming language above for Project 3? Explain briefly (1 to 2 sentences).

We chose Java as our programming language because it has robust support for String methods, allowing easier string manipulation. In addition to this, the lectures in class on binary search trees were implemented using C, which we initially implemented this program with, and eventually translated it into Java to make it much more efficient.

3. Depending on the programming language used:

a. List the libraries or APIs that you used in your implementation

Java Standard Libraries

- Java.io.*
- java.util.Scanner
- java.util.regex.Matcher
- java.util.regex.Pattern
- java.util.*

b. Indicate how to compile (if it is a compiled language) your codes, and how RUN (execute) your program from the COMMAND LINE. Examples are shown below highlighted in yellow. Replace them accordingly. Make sure that all your group members test what you typed below because I will follow them verbatim. I will initially test your solution using the sample input text file that you submitted. Thereafter, I will run it again using my own test data:

- How to compile from the command line (for compiled language only):

```
C:\CCDSALG>javac MainDriver.java
```

- How to run from the command line

```
C:\CCDSALG>java MainDriver
```

4. How did you implement your BST data structure? Did you implement a single BST or multiple BST? Why? Explain briefly (2 to 3 sentences).

We implemented a common recursive BST data structure that includes the data, and the left and right child. The data is stored as a string while the left and right child subtrees are also recursions of the same kind of BST data structure. We also used a single BST to store all the data since our Insert function contains the algorithm to alphabetically arrange the nodes.

5. Disclose what is NOT working correctly in your solution. Be honest about this. Explain briefly the reason why your group was not able to make it work.

To our knowledge, the program is fully functional with no known cases of failure.

6. What do you think is the level of difficulty of the project (was it easy, medium or hard)? Which part is hard (if you answered hard)? Type your answer individually for this question.

Gan, John: The project was easy to do, since the scale of the project is relatively small and not very complex, and implementing the program in Java also makes it much smore efficient in terms of code complexity.

Noblefranca, Noel: The project was easy, as translating our original C implementation into Java required only slight modifications to the algorithms and using Java makes it easier to process the data.

Remudaro, Angelos: This project was on the easy side because of how simple BSTs are implemented. The BST functions are also not that difficult and the built in functions of Java made splitting the strings a lot easier.

7. Fill-up the table below. Refer to the rubric in the project specs. It is suggested that you do first an individual self-assessment. Thereafter, compute the average evaluation for your group, and encode it below.

REQUIREMENT	**AVE. OF SELF-ASSESSMENT**
1. BST	47 (max. 50 points)
2. Input File Parsing	20 (max. 20 points)
3. Output File	13 (max. 15 points)
5. Documentation	9 (max. 10 points)
6. Compliance with Instructions	5 (max. 5 points)

TOTAL SCORE 94 over 100.

NOTE: The evaluation that the instructor will give is not necessarily going to be the same as what you indicated above. The self-assessment serves primarily as a guide.