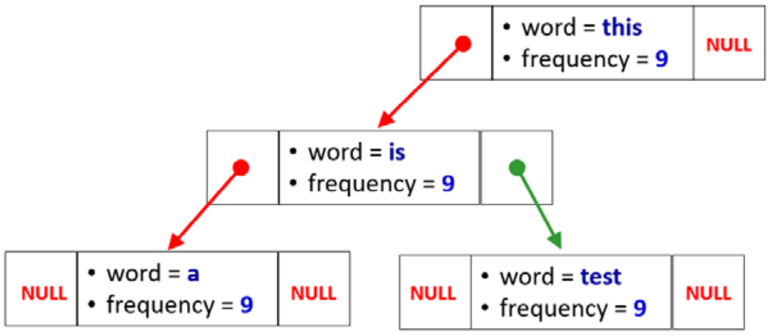
CPSC 319 – Summer 2020 Assignment 2 **Binary Search Trees** (10% of total final grade)	**Released:**	July 16 (Thursday) @ 1:00 PM
	Due:	August 2 (Sunday) @ 11:59 PM
	# days to complete:	17.5 days

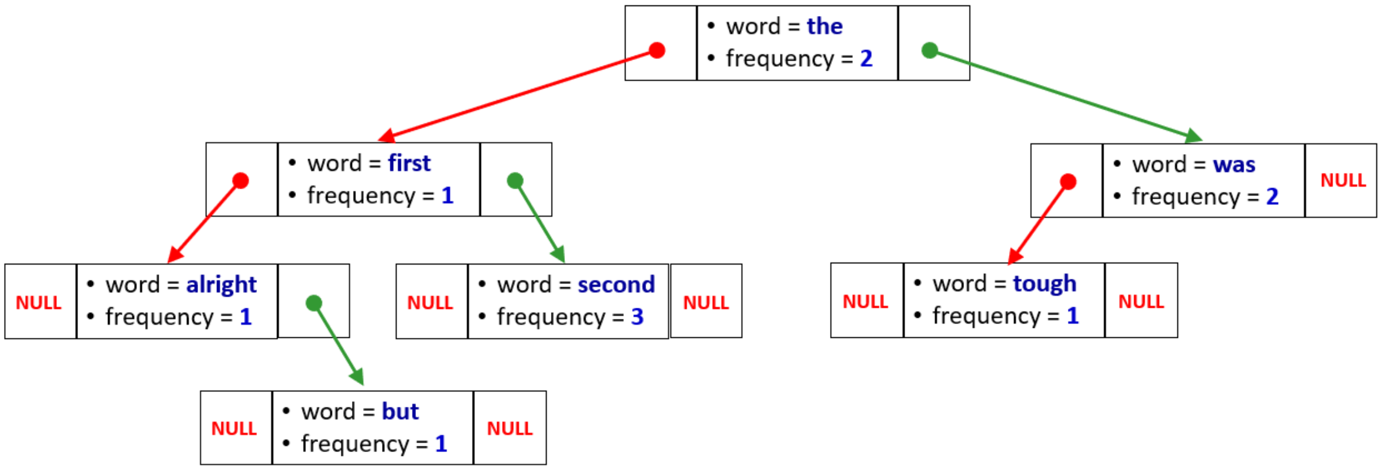
GOAL

Trees are invaluable data structures. In this assignment, you will write the code that will create a binary search tree (BST) of unique words from a given input text file, and also keep track of the frequency of the words in the text. For example:

Example input text file "example01.txt"	**Resulting BST**

`This is a test. This is a test.`
`This is a test. This is a test.`
`This is a test. This is a test.`
`This is a test. This is a test.`
`This is a test.`
	 word = this
	frequency = 9
	NULL
	word = is
	frequency = 9
	NULL
	word = a
	frequency = 9
	NULL
	NULL
	word = test
	frequency = 9
	NULL

Example Input text file "example02.txt"
`The first second was alright, but the second second was tough.`
Resulting BST

 word = the
frequency = 2
word = first
frequency = 1
word = was
frequency = 2
NULL
NULL
word = alright
frequency = 1
NULL
word = second
frequency = 3
NULL
NULL
word = tough
frequency = 1
NULL
NULL
word = but
frequency = 1
NULL

The nodes from the BST store the word itself and its frequency in the input text file. Using this structure for a tree, it is possible to find if a word occurs in the text in O($\log_2 n$) (i.e., if the tree is of minimum height) as well as the number of times the word occurs in the text.

Specifications for reading and processing the input text file:

- All text will be considered to be lower-case, even if the word in the file has an upper-case character as the first character, or even if it is all upper-case, or a mixture of upper- and lower-case.
- Words are delimited with spaces.
- All punctuation will be ignored.
- Hyphenated words will be considered as two words (or state another way, the hyphen is a delimiter).

INSTRUCTIONS

Write the Java program that will do the following:

1. **Request the user for the name of a text file.** This should be an ASCII text file (i.e., like the format from assignment #1)

Example (display screen) for "example01.txt" file:

```
> Enter the input file name:  example01
```

2. **Use the words from 2 input files** (available at the assignment's D2L page) **to create the BST specified in the previous page.** You will have to do the conversions to lower-case and take care of all of the other characters that do not make up the word as defined above.

   Two Input files: text1 (i.e., merged example01 and example02 from this assignment specs)**, text2** (i.e., an excerpt from "The Rime of the Ancient Mariner in Seven Parts" by Samuel Taylor Coleridge)

3. **When the tree has been created, it should supply the following information as a display output:**

(3.1) **The total number of words in the file.**
Use either IN-ORDER, PRE-ORDER, or POST-ORDER traversal, counting the number of times you visit each node in the BST.

(3.2) **The number of unique words in the file.**
Use either IN-ORDER, PRE-ORDER, or POST-ORDER traversal, counting the number of nodes visited containing the word with frequency = 1

(3.3) **The word which occurs most often and the number of times that it occurs.**
Use either IN-ORDER, PRE-ORDER, or POST-ORDER traversal, looking for the word(s) with the largest frequency.

Example (display screen for question #3) for example01 using POST-ORDER traversal (i.e., left, right, node):

```
> Total number of words in example01 = 4

> Number of unique words in example01 =  0

> The word(s) which occur(s) most often and the number of times that it/they occur(s) =
   a = 9 times
   test = 9 times
   is = 9 times
   this = 9 times
```

Example (display screen for question #3) for example02 using POST-ORDER traversal (i.e., left, right, node):

```
> Total number of words in example02 = 7

> Number of unique words in example02 =  4

> The word(s) which occur(s) most often and the number of times that it/they occur(s) =
   second = 3 times
```

4. The user should also be able to request information about any word that is input when requested.

The result should be whether the word exists, and the number of times it appears (i.e., its frequency). Use either IN-ORDER, PRE-ORDER, or POST-ORDER traversal.

Example (display screen) for example01:

```
> Enter the word you are looking for in example01 ? hello
> Word not found!

> Enter the word you are looking for in example01 ? is
> Found! It appears 9 times in the input text file
```

5. BONUS – The user should also be able to request to display the entire tree using any of the 3 traversal methods.

The result should be each word in the tree separated by a single space.

Example (display screen for question #5) for example01:

```
> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 1
> IN-ORDER output: a is test this

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 2
> PRE-ORDER output: this is a test

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 3
> POST-ORDER output: a test is this
```

Example (display screen for question #5) for example02:

```
> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example02 ? 1
> IN-ORDER output: alright but first second the tough was

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example02 ? 2
> PRE-ORDER output: the first alright but second was tough

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example02 ? 3
> POST-ORDER output: but alright second first tough was the
```

So, here is the sequence of what should be displayed by running your program for example 01

```
> Enter the input file name:  example01
```

```
> Total number of words in example01 = 4

> Number of unique words in example01 =  0

> The word(s) which occur(s) most often and the number of times that it/they occur(s) =
   a = 9 times
   test = 9 times
   is = 9 times
   this = 9 times
```

For this next display, you should keep prompting the user using any key to finish it

```
> Enter the word you are looking for in example01 ? hello
> Word not found!

> Enter the word you are looking for in example01 ? is
> Found! It appears 9 times in the input text file
```

Finally, the last display

```
> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 1
> IN-ORDER output: a is test this

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 2
> PRE-ORDER output: this is a test

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 3
> POST-ORDER output: a test is this
```

HAND-IN

1. Your code + README (w/ instructions on how to run and use/interact with your program)
2. Include all of the above items in a zipped folder titled (asgmt2-your LAST NAME-ID).zip and submit it to D2L dropbox.
3. LATE ASSIGNMENTS WILL NOT BE ACCEPTED

MARKING

Source code that does not compile or produces run-time errors will receive a grade of 0%.

	Item	Points
1	**Request the user for the name of a text file** (item #2, page 2 of this document)	**1 point**
2	**When the tree has been created, execute, and display the information** (items 3.1, 3.2, 3.3, page 2 of this document)	**6 points** 1 point for each of the 3 items (3.1, 3.2, 3.3) for each of the 2 input files (text1, text2) – i.e., 3 x 2 = 6 points
2	**The user should also be able to request information about any word that is input when requested** (item #4, page 3 of this document)	**4 points** 2 points (1 for the user prompt interface + 1 point for the correct answer from your program) for each of the 2 input files (text1, text2) – i.e., 2 x 2 = 4 points
3	**Output for each of the 3 traversal modes** (page 4 of this document)	**6 BONUS points** 1 point for the correct output of each of the 3 traversal modes for each of the 2 input files (text1, text2) – i.e., 3 x 2 = 6 bonus points
	TOTAL	**11 points (**10% of the assignment grade) **+ 6 BONUS points** Note: I will distribute the bonus points as follows: 1 bonus point to quiz #1, 1 bonus point to quiz #2 and 4 bonus points to the final exam

INDIVIDUAL WORK

- The assignment must be done **individually,** so you must write up the solutions *on your own* in *your own words*.
- Everything that you hand in must be your original work, except for the code copied from the textbook, lecture material (i.e., slides, notes), web, or that supplied by your TA. When someone else's code is used like this, you **must** acknowledge the source explicitly, citing the sources in a scientific way (i.e., including author(s), title, page numbers, URLs, etc.).
- Copying another student's work constitutes academic misconduct, a severe offence that will be dealt with rigorously in all cases. Please read the sections of the University Calendar under the heading "Student Misconduct."
- If you are in doubt whether a specific form of aid is allowed, ask your instructor!
- Contact your TA if you have problems getting your code to work.
- Note that your code may be checked thoroughly for plagiarism by computer.

END OF THE ASSIGNMENT