COP 3502C Spring 2019
Final term Assignment 1
Total points: 15

Introduction: For this assignment you have to write a c program that will utilize the Binary Search Tree (BST).

What should you submit?
Write all the code in a single file and upload the .c file.

Please include the following lines in the beginning of your code. By writing this line you also agree that you have written the code:

/* COP 3502C Final term Assignment 2
This program is written by: Your Full Name */

Compliance with Rules: UCF Golden rules apply towards this assignment and submission. Assignment rules mentioned in the syllabus, are also applied in this submission.

Problem

In this assignment, you have to read a text file (in.txt) that contains a set of words. The first line of the file contains 3 numbers (N, S, D). These numbers represent the sequence of input words in your file.

N: represents the number of words to read to build binary search tree. You have to write a recursive insert code to create and insert these words into the binary search tree. After inserting all the items, you should show the words in Pre order, In order, and Post order. So, you need to create three functions for this purpose.

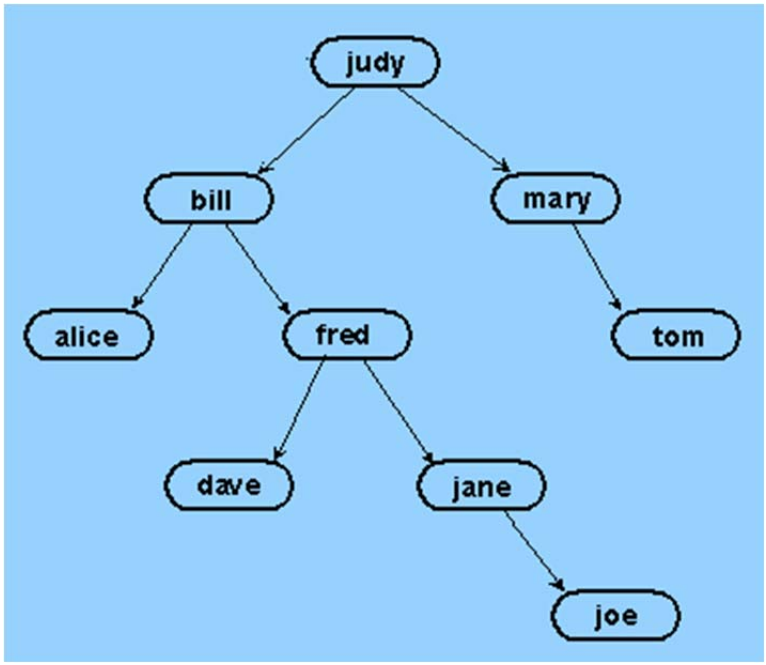
S: represents the number of the words to search from the tree. These S words are placed after the first N words in the input file.You need to implement a search function that will be able to search these words in your BST.

//the following paragraph is updated compared to the last version

Additionally, the search words will also be used to look in the binary search tree and count number of words in the tree that come before the search word, alphabetically. You will need to create a recursive function with the following prototype ***CountBefore(treeNode* root, char searchKey[]),*** where treeNode is the node structure of your tree.

D: represent the number of words to delete from the BST. This list of words are placed after N+S words in the input file. Write a recursive delete function for your task. After deleting all the items, also show the tree in three different orders of traversals.

The figure below shows an example of BST with strings.



All the output must be written in the out.txt file according the sample output shown bellow. You may output in the console, however, we will mainly see the out.txt file for our given input file.

Example

Sample Input file in.txt:

```
9 3 4
judy
mary
bill
alice
tom
fred
jane
joe
dave
jane
jones
judy
alice
mary
judy
fred
```

Sample Output file out.txt:

```
Pre Order: judy bill alice fred dave jane joe mary tom
In Order: alice bill dave fred jane joe judy mary tom
Post Order: alice dave joe jane fred bill tom mary judy
Search Phase:
jane: Found, Items before: 4 //it is corrected compared to last version
jones: Not found, Items before: 0
judy: Found, Items before: 6 //it is corrected compared to last version
Delete Phase:
alice: deleted
mary: deleted
judy: deleted
fred: deleted
Pre Order: tom bill jane dave joe  //depending on your delete operation it might change
In Order: bill dave jane joe tom
Post Order: dave joe jane bill tom //depending on your delete operation it might change
```

Requirement:

You must use file IO, Binary Search Tree and relevant traversal for your solution.

Your program must compile in code blocks GNU GCC Compler setting.

Rubric:

1) Implementing insertion: 4.5 (-0.5 for each test case. 9 test cases)
2) In order, postorder, and pre-order: 1.5 (-0.5 for each function)
3) Implementing search: 2 (-0.66 each test case. 3 test cases)
4) Implementing counting count below: 2 (-0.66 each test case 3 test cases)
5) Implementing Delete: 5 (-1.25 for each test case. 4 test cases)

Hint: use the strcmp() function of string.h to compare strings. Example strcmp() code bellow.

```
#include <stdio.h>
#include <string.h>

int main()
{
   char str1[] = "abcd", str2[] = "abce", str3[] = "bcde", str4[]="abCd", str5[]="abcd";
   int result;

   // comparing strings str1 and str2
   result = strcmp(str1, str2);
   printf("strcmp(str1, str2) = %d\n", result);

   result = strcmp(str2, str1);
   printf("strcmp(str2, str1) = %d\n", result);

   // comparing strings str1 and str3
   result = strcmp(str1, str3);
   printf("strcmp(str1, str3) = %d\n", result);

   // comparing strings str1 and str3
   result = strcmp(str3, str1);
   printf("strcmp(str3, str1) = %d\n", result);

      // comparing strings str1 and str2
   result = strcmp(str1, str4);
   printf("strcmp(str1, str4) = %d\n", result);

   result = strcmp(str4, str1);
   printf("strcmp(str4, str1) = %d\n", result);

    result = strcmp(str1, str5);
   printf("strcmp(str1, str5) = %d\n", result);

   return 0;
}

/*output
strcmp(str1, str2) = -1
strcmp(str2, str1) = 1
strcmp(str1, str3) = -1
strcmp(str3, str1) = 1
strcmp(str1, str4) = 1
strcmp(str4, str1) = -1
strcmp(str1, str5) = 0
*/
```