ASSIGNMENT-3
DATA STRUCTURES
(CSU33D05)

SUBMITTED TO:
Dr. Mélanie Bouroche

SUBMITTED BY:
PRACHI SINGHROHA
(Student ID: 21355131)

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Declaration concerning plagiarism

I have read and I understand the plagiarism provisions in the General Regulations of the *University Calendar* for the current year, found at http://www.tcd.ie/calendar

I have completed the Online Tutorial in avoiding plagiarism 'Ready, Steady, Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write

STUDENT NUMBER: 21355131

SIGNED:

DATE: 19.11.2021

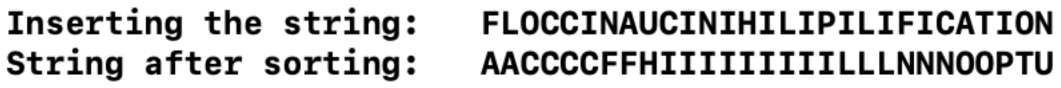
Task 1:

I have used the skeleton given on the blackboard. I have updated the following functions there

1. void tree_insert (Tree_Node** root, char data);
2. Tree_Node* create_bst (char data[]);
3. Tree_Node* tree_search (Tree_Node* root, char data);
4. void tree_print_sorted (Tree_Node* root);
5. void tree_delete (Tree_Node* root);

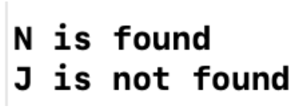
The code does

a. Print the string "FLOCCINAUCINIHILIPILIFICATION" in sorted order



```
Inserting the string:   FLOCCINAUCINIHILIPILIFICATION
String after sorting:   AACCCCFFHIIIIIIIIILLLNNNOOPTU
```

b. Accurately reports whether a given letter is in the tree using the *tree_search* function

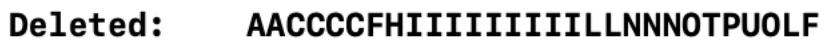


```
N is found
J is not found
```

(P.S. I checked for random characters while running my code)

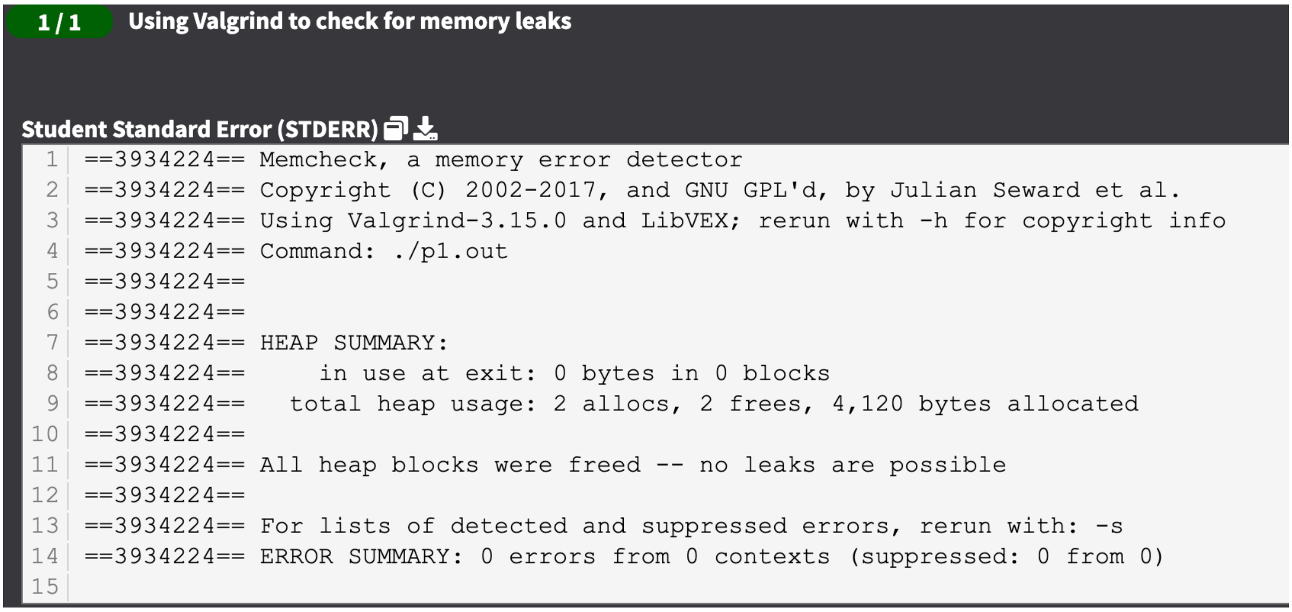
c. Fully delete the tree and free all allocated memory using the *tree_delete* function

From my editor:



```
Deleted:    AACCCCFHIIIIIIIIILLNNNOTPUOLF
```

From submitty:



1 / 1
Using Valgrind to check for memory leaks
Student Standard Error (STDERR)

```
==3934224== Memcheck, a memory error detector
==3934224== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3934224== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3934224== Command: ./p1.out
==3934224==
==3934224==
==3934224== HEAP SUMMARY:
==3934224==     in use at exit: 0 bytes in 0 blocks
==3934224==   total heap usage: 2 allocs, 2 frees, 4,120 bytes allocated
==3934224==
==3934224== All heap blocks were freed -- no leaks are possible
==3934224==
==3934224== For lists of detected and suppressed errors, rerun with: -s
==3934224== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Task 2:

I created the binary search tree and used the functions to complete the task given with no errors.

The output is as follows:

```
Generating 110462 books... OK

Profiling listdb
-------------------------------------------

Total Inserts                :      110462
Num Insert Errors            :           0
Avg Insert Time              :   0.000005 s
Var Insert Time              :   0.000001 s
Total Insert Time            :   1.094936 s

Total Title Searches         :       11046
Num Title Search Errors      :           0
Avg Title Search Time        :   0.001079 s
Var Title Search Time        :   0.004696 s
Total Title Search Time      :  12.001198 s

Total Word Count Searches    :       11046
Num Word Count Search Errors :           0
Avg Word Count Search Time   :   0.000956 s
Var Word Count Search Time   :   0.004850 s
Total Word Count Search Time :  10.634808 s

STAT
Avg comparisons per search  -> 55545.572470
List size matches expected? -> Y

Profiling bstdb
-------------------------------------------

```

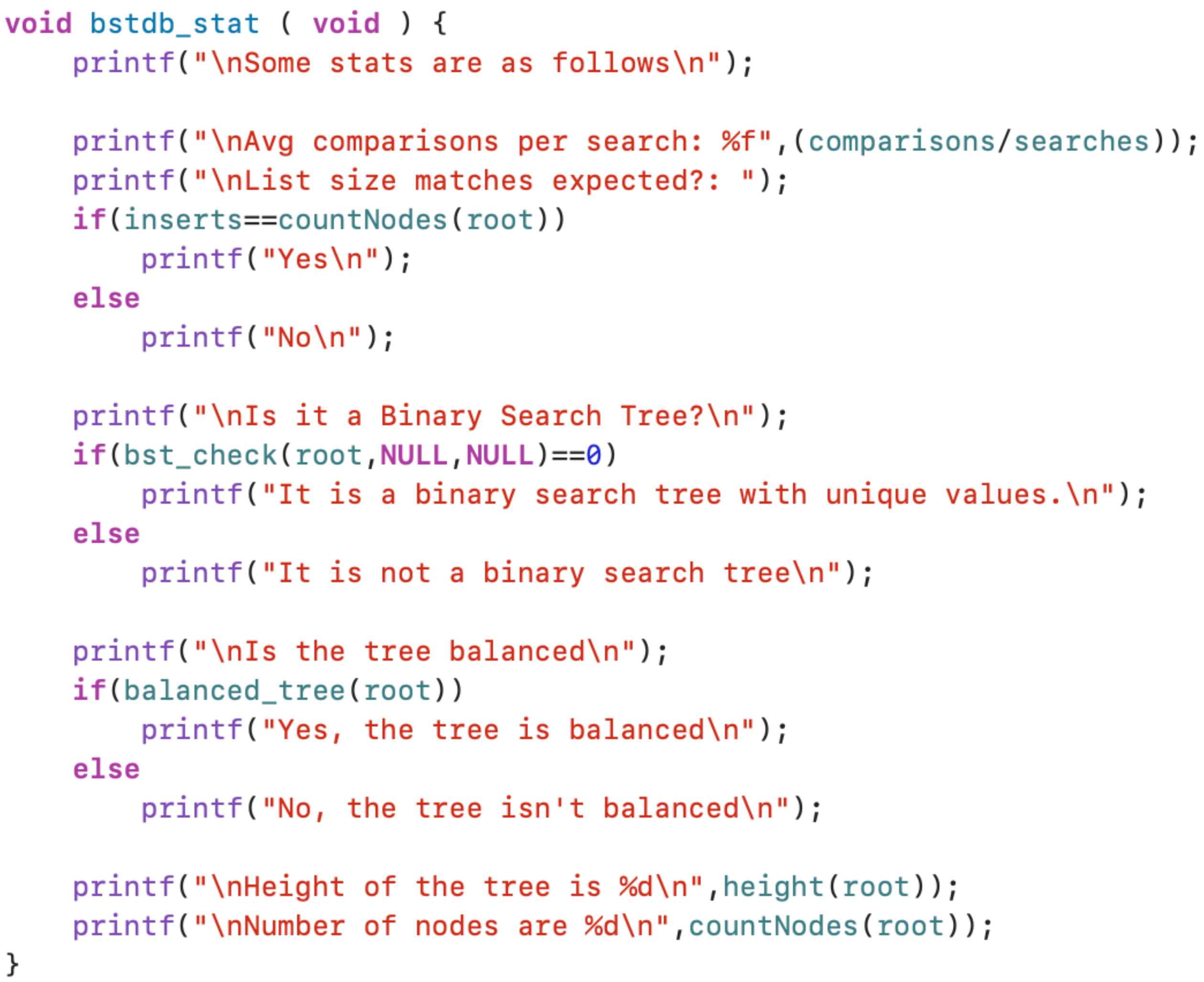
```
Total Inserts                : 	  110462
Num Insert Errors            : 	       0
Avg Insert Time              :   0.000007 s
Var Insert Time              :   0.000002 s
Total Insert Time            :   1.316130 s

Total Title Searches         :        11046
Num Title Search Errors      :            0
Avg Title Search Time        :   0.000008 s
Var Title Search Time        :   0.000000 s
Total Title Search Time      :   0.137125 s

Total Word Count Searches    :        11046
Num Word Count Search Errors :            0
Avg Word Count Search Time   :   0.000008 s
Var Word Count Search Time   :   0.000001 s
Total Word Count Search Time :   0.152257 s


Some stats are as follows

Avg comparisons per search: 17.068396
List size matches expected?: Yes

Is it a Binary Search Tree?
It is a binary search tree with unique values.

Is the tree balanced
Yes, the tree is balanced

Height of the tree is 20

Number of nodes are 110462


Press Enter to quit...
```

```
Generating 104977 books... OK

Profiling listdb
-------------------------------------------

Total Inserts                :       104977
Num Insert Errors            :            0
Avg Insert Time              :   0.000005 s
Var Insert Time              :   0.000002 s
Total Insert Time            :   1.073109 s

Total Title Searches         :        10497
Num Title Search Errors      :            0
Avg Title Search Time        :   0.001056 s
Var Title Search Time        :   0.004801 s
Total Title Search Time      :  11.169333 s

Total Word Count Searches    :        10497
Num Word Count Search Errors :            0
Avg Word Count Search Time   :   0.001028 s
Var Word Count Search Time   :   0.004384 s
Total Word Count Search Time :  10.871674 s

STAT
Avg comparisons per search  -> 52327.035105
List size matches expected? -> Y

Profiling bstdb
-------------------------------------------

Total Inserts                :       104977
Num Insert Errors            :            0
Avg Insert Time              :   0.000007 s
Var Insert Time              :   0.000002 s
Total Insert Time            :   1.083178 s

Total Title Searches         :        10497
Num Title Search Errors      :            0
Avg Title Search Time        :   0.000007 s
Var Title Search Time        :   0.000000 s
Total Title Search Time      :   0.127576 s

Total Word Count Searches    :        10497
Num Word Count Search Errors :            0
Avg Word Count Search Time   :   0.000007 s
Var Word Count Search Time   :   0.000000 s
Total Word Count Search Time :   0.122933 s


Some stats are as follows

Avg comparisons per search: 16.987377
List size matches expected?: Yes

Is it a Binary Search Tree?
It is a binary search tree with unique values.

Is the tree balanced
Yes, the tree is balanced

Height of the tree is 20

Number of nodes are 104977


Press Enter to quit...
```

To keep the tree balanced I created an AVL binary search tree because

i. It is a self balancing tree
ii. The heights of the two child subtrees of any node differ by at most by 1
iii. If at any time there is a height difference more than 1, rebalancing takes place
iv. Used extensively in database applications where frequent lookups for data required.

My *bstdb_stat* function is as follows:



```
void bstdb_stat ( void ) {
    printf("\nSome stats are as follows\n");

    printf("\nAvg comparisons per search: %f",(comparisons/searches));
    printf("\nList size matches expected?: ");
    if(inserts==countNodes(root))
        printf("Yes\n");
    else
        printf("No\n");

    printf("\nIs it a Binary Search Tree?\n");
    if(bst_check(root,NULL,NULL)==0)
        printf("It is a binary search tree with unique values.\n");
    else
        printf("It is not a binary search tree\n");

    printf("\nIs the tree balanced\n");
    if(balanced_tree(root))
        printf("Yes, the tree is balanced\n");
    else
        printf("No, the tree isn't balanced\n");

    printf("\nHeight of the tree is %d\n",height(root));
    printf("\nNumber of nodes are %d\n",countNodes(root));
}
```

It gives the following stats:

1. Average comparisons per search
2. Whether or not the list size matches the expected
3. Whether or not it is binary search tree
4. Whether or not the tree is balanced
5. Height of the tree
6. Number of nodes in the tree

References:

1. Skeleton code provided
2. Slides provided on blackboard
3. Searched BST and AVLBST on google and took refrences from random websites, a few of them are as follows:
a. GeeksForGeeks
b. SetScholars
c. CodesDope
4. Jacob Sorber: Understand and Implement a Binary Search Tree in C (Youtube)