



# DATA STRUCTURES

## ASSIGNMENT 3

NAME - SUGANDHI SAGGU

TCD ID – 21355223

### Task 1:

Student STDOUT.txt 	Expected STDOUT.txt 
<pre>1 Raw Data: 2 FLOCCINAUCINIHIILIPILIFICATION 3 4 Sorted Data: 5 AACCCCFHHIIIIIIILLNNOOPTU 6 7 A found 8 B is not in the dataset 9 C found 10 D is not in the dataset 11 E is not in the dataset 12 F found 13 G is not in the dataset 14 H found 15 I found 16 J is not in the dataset 17 K is not in the dataset 18 L found 19 M is not in the dataset 20 N found 21 O found 22 P found 23 Q is not in the dataset 24 R is not in the dataset 25 S is not in the dataset 26 T found 27 U found 28 V is not in the dataset 29 W is not in the dataset 30 X is not in the dataset 31 Y is not in the dataset 32 Z is not in the dataset 33</pre>	<pre>1 Raw Data: 2 FLOCCINAUCINIHIILIPILIFICATION 3 4 Sorted Data: 5 AACCCCFHHIIIIIIILLNNOOPTU 6 7 A found 8 B is not in the dataset 9 C found 10 D is not in the dataset 11 E is not in the dataset 12 F found 13 G is not in the dataset 14 H found 15 I found 16 J is not in the dataset 17 K is not in the dataset 18 L found 19 M is not in the dataset 20 N found 21 O found 22 P found 23 Q is not in the dataset 24 R is not in the dataset 25 S is not in the dataset 26 T found 27 U found 28 V is not in the dataset 29 W is not in the dataset 30 X is not in the dataset 31 Y is not in the dataset 32 Z is not in the dataset 33</pre>

#### 1/1 Using Valgrind to check for memory leaks

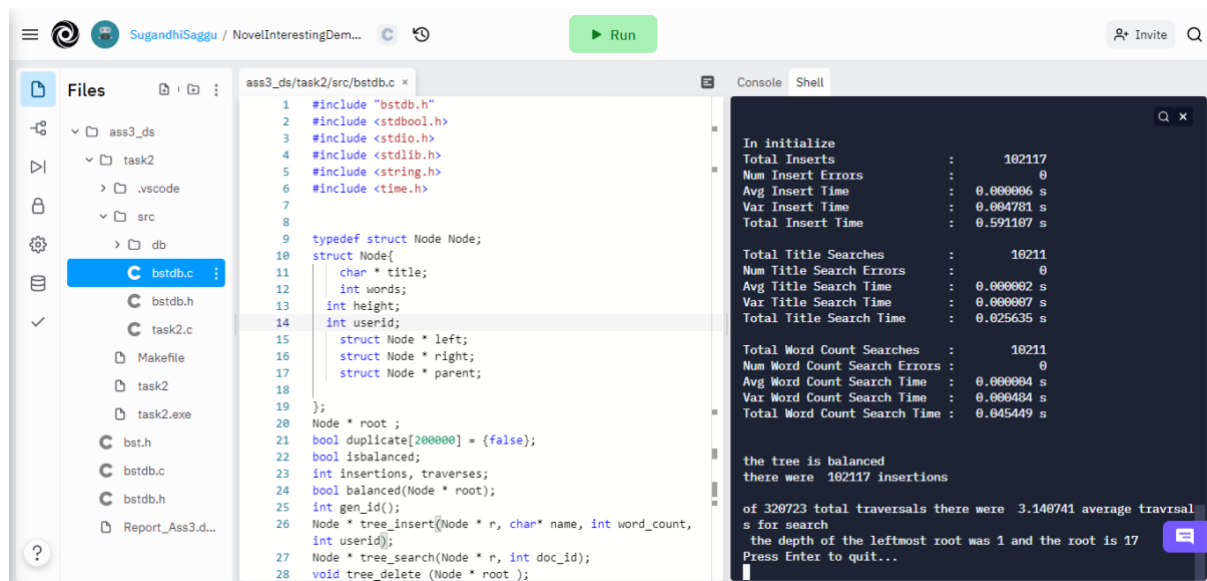
[Hide Details](#)

[Visualize whitespace characters](#)

##### Student Standard Error (STDERR)

```
1 ==1253251== Memcheck, a memory error detector
2 ==1253251== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
3 ==1253251== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
4 ==1253251== Command: ./pl.out
5 ==1253251==
6 ==1253251==
7 ==1253251== HEAP SUMMARY:
8 ==1253251==      in use at exit: 0 bytes in 0 blocks
9 ==1253251==    total heap usage: 30 allocs, 30 frees, 4,792 bytes allocated
10 ==1253251==
11 ==1253251== All heap blocks were freed -- no leaks are possible
12 ==1253251==
13 ==1253251== For lists of detected and suppressed errors, rerun with: -s
14 ==1253251== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
15
```

## Task 2:



The screenshot shows a C++ IDE with a file explorer on the left, a code editor in the center, and a console on the right. The file explorer shows a project structure with folders 'ass3\_ds', 'task2', and 'src', and files 'bst.h', 'bstdb.c', 'bstdb.h', 'task2.c', 'task2.exe', and 'Report\_Ass3.d...'. The code editor displays the implementation of an AVL tree in 'bstdb.c'. The console shows the output of the program, including statistics for insertions and searches, and a confirmation that the tree is balanced.

```
1 #include "bstdb.h"
2 #include <stdbool.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #include <time.h>
7
8 typedef struct Node Node;
9
10 struct Node{
11     char * title;
12     int words;
13     int height;
14     int userid;
15     struct Node * left;
16     struct Node * right;
17     struct Node * parent;
18 }
19
20 Node * root ;
21 bool duplicate[200000] = {false};
22 bool isbalanced;
23 int insertions, traverses;
24 bool balanced(Node * root);
25 int gen_id();
26 Node * tree_insert(Node * r, char* name, int word_count,
27 int userid);
28 Node * tree_search(Node * r, int doc_id);
29 void tree_delete (Node * root );
```

Console Output:

```
In initialize
Total Inserts : 102117
Num Insert Errors : 0
Avg Insert Time : 0.000006 s
Var Insert Time : 0.004781 s
Total Insert Time : 0.591107 s

Total Title Searches : 10211
Num Title Search Errors : 0
Avg Title Search Time : 0.000002 s
Var Title Search Time : 0.000007 s
Total Title Search Time : 0.025635 s

Total Word Count Searches : 10211
Num Word Count Search Errors : 0
Avg Word Count Search Time : 0.000004 s
Var Word Count Search Time : 0.000484 s
Total Word Count Search Time : 0.045449 s

the tree is balanced
there were 102117 insertions

of 320723 total traversals there were 3.148741 average traversal
s for search
the depth of the leftmost root was 1 and the root is 17
Press Enter to quit...
```

In this data structures, I have used AVL Tree. It is a self balancing binary tree. In an AVL tree, the height of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property. The height of an AVL tree is always  $O(\log(n))$ , where  $n$  is the number of nodes.