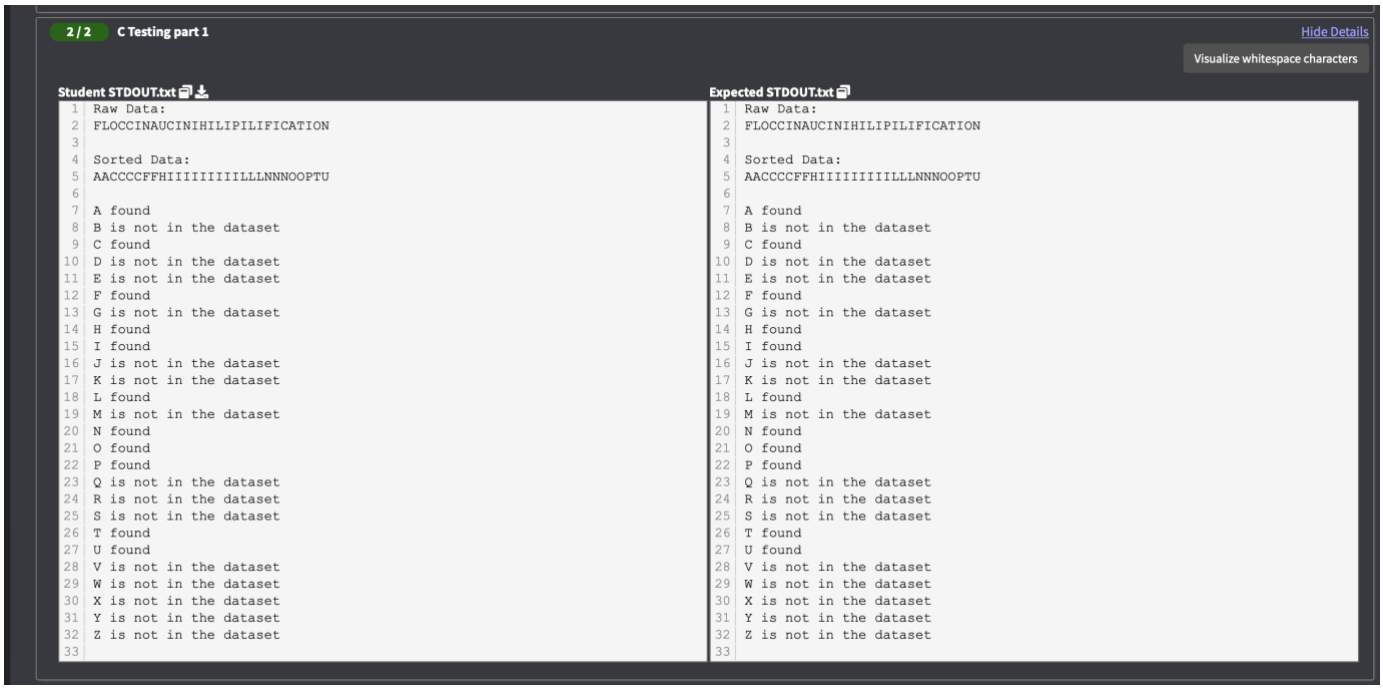
ASSIGNMENT 3

Name: Vanaj Kamboj
Roll No: 21355100

Task 1

Output on Submitty



2 / 2 C Testing part 1

Hide Details

Visualize whitespace characters

Student STDOUT.txt

```
Raw Data:
FLOCCINAUCINIHILIPILIFICATION

Sorted Data:
AACCCCFFHIIIIIIIIIILLLNNNOOPTU

A found
B is not in the dataset
C found
D is not in the dataset
E is not in the dataset
F found
G is not in the dataset
H found
I found
J is not in the dataset
K is not in the dataset
L found
M is not in the dataset
N found
O found
P found
Q is not in the dataset
R is not in the dataset
S is not in the dataset
T found
U found
V is not in the dataset
W is not in the dataset
X is not in the dataset
Y is not in the dataset
Z is not in the dataset

```

Expected STDOUT.txt

```
Raw Data:
FLOCCINAUCINIHILIPILIFICATION

Sorted Data:
AACCCCFFHIIIIIIIIIILLLNNNOOPTU

A found
B is not in the dataset
C found
D is not in the dataset
E is not in the dataset
F found
G is not in the dataset
H found
I found
J is not in the dataset
K is not in the dataset
L found
M is not in the dataset
N found
O found
P found
Q is not in the dataset
R is not in the dataset
S is not in the dataset
T found
U found
V is not in the dataset
W is not in the dataset
X is not in the dataset
Y is not in the dataset
Z is not in the dataset

```

1 / 1 Using Valgrind to check for memory leaks

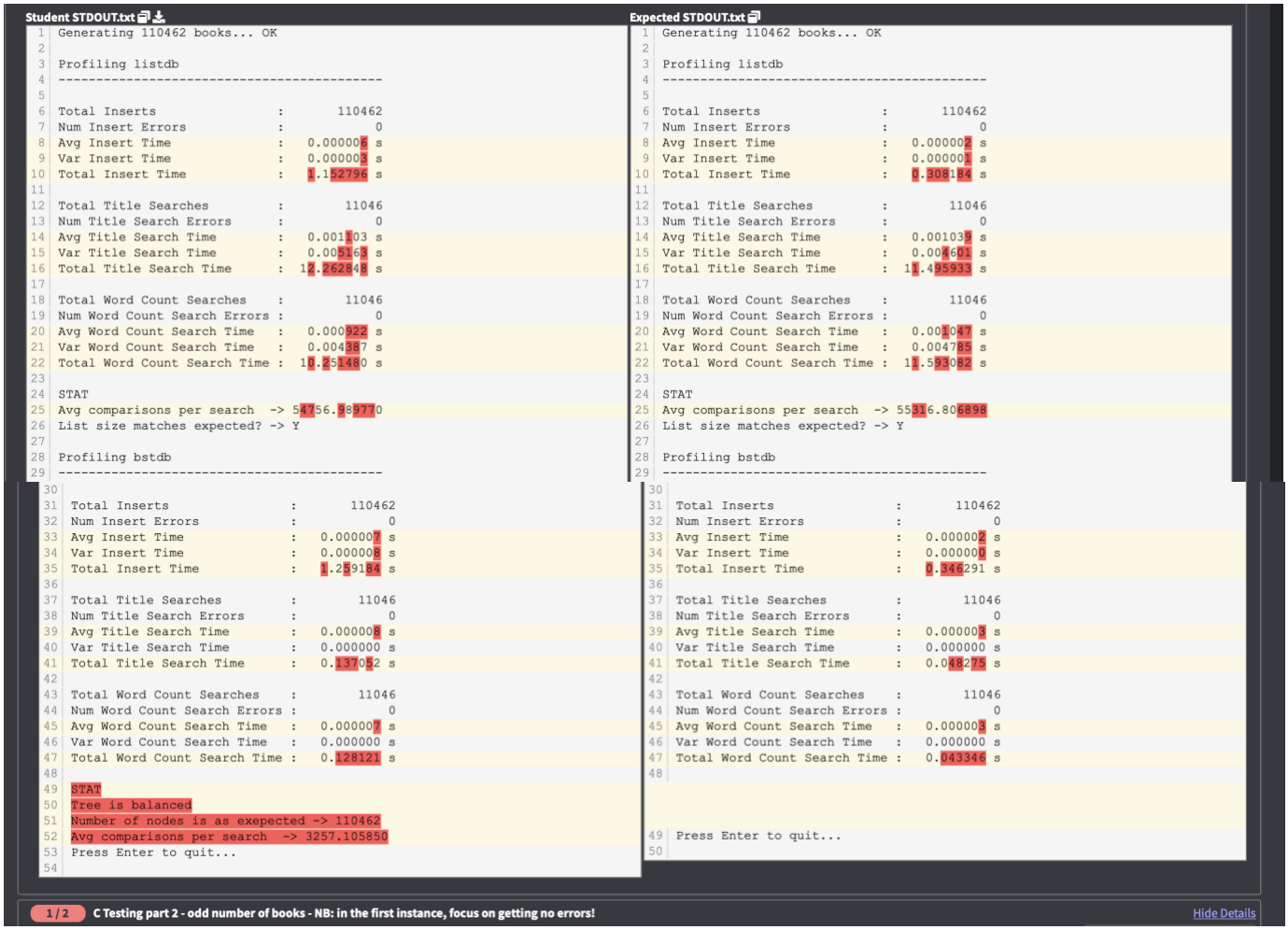
Hide Details

Visualize whitespace characters

Student Standard Error (STDERR)

```
==1144016== Memcheck, a memory error detector
==1144016== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1144016== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==1144016== Command: ./p1.out
==1144016==
==1144016==
==1144016== HEAP SUMMARY:
==1144016==     in use at exit: 0 bytes in 0 blocks
==1144016==   total heap usage: 30 allocs, 30 frees, 4,792 bytes allocated
==1144016==
==1144016== All heap blocks were freed -- no leaks are possible
==1144016==
==1144016== For lists of detected and suppressed errors, rerun with: -s
==1144016== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

Task 2



Student STDOUT.txt

```
Generating 110462 books... OK

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

Total Inserts                :       110462
Num Insert Errors            :            0
Avg Insert Time              :     0.000006 s
Var Insert Time              :     0.000003 s
Total Insert Time            :     1.152796 s

Total Title Searches         :        11046
Num Title Search Errors      :            0
Avg Title Search Time        :     0.001103 s
Var Title Search Time        :     0.005163 s
Total Title Search Time      :    12.262848 s

Total Word Count Searches    :        11046
Num Word Count Search Errors :            0
Avg Word Count Search Time   :     0.000922 s
Var Word Count Search Time   :     0.004387 s
Total Word Count Search Time :    10.251480 s

STAT
Avg comparisons per search  -> 54756.989770
List size matches expected? -> Y

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

Total Inserts                :       110462
Num Insert Errors            :            0
Avg Insert Time              :     0.000007 s
Var Insert Time              :     0.000008 s
Total Insert Time            :     1.259184 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      :     0.137052 s

Total Word Count Searches    :        11046
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 :     0.128121 s

STAT
Tree is balanced
Number of nodes is as exepected -> 110462
Avg comparisons per search  -> 3257.105850
Press Enter to quit...
```

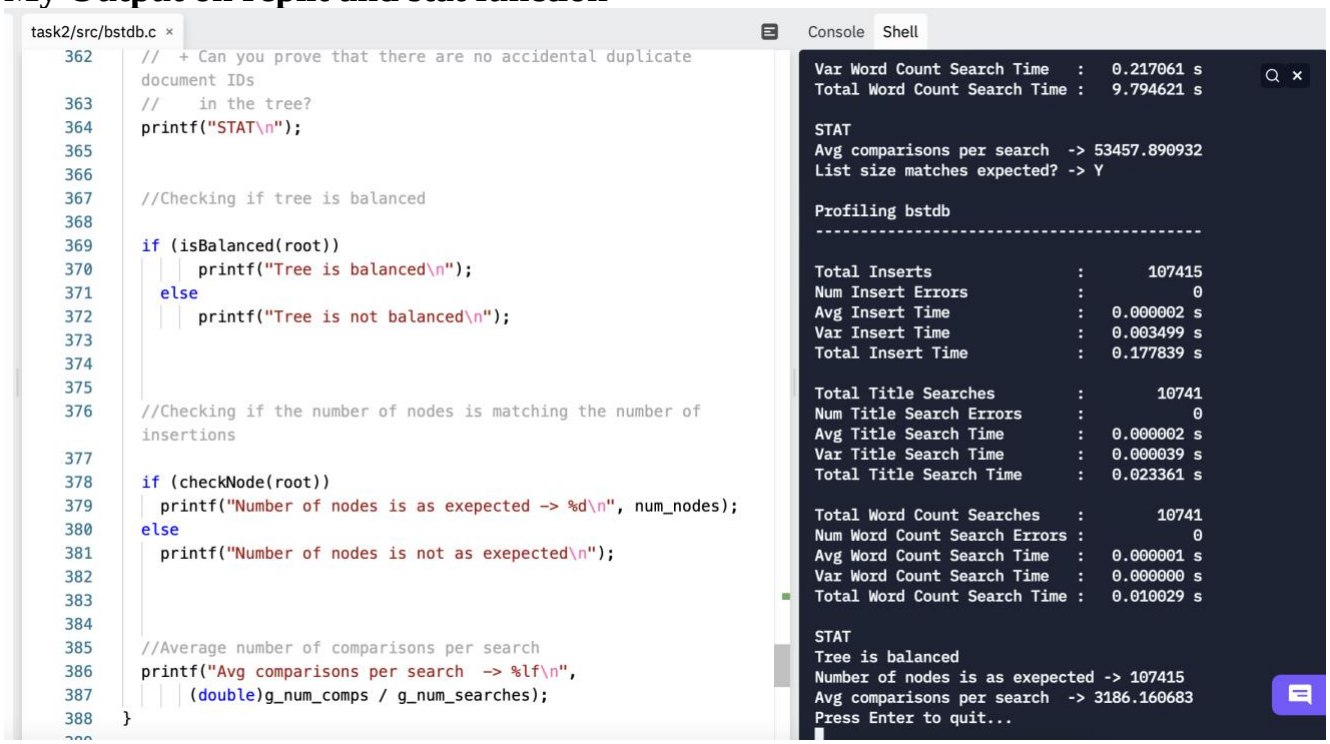
Expected STDOUT.txt

```
Generating 110462 books... OK

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

Total Inserts                :       110462
Num Insert Errors            :            0
Avg Insert Time              :     0.000002 s
Var Insert Time              :     0.000001 s
Total Insert Time            :     0.308184 s

Total Title Searches         :        11046
Num Title Search Errors      :            0
Avg Title Search Time        :     0.001039 s
Var Title Search Time        :     0.004601 s
Total Title Search Time      :    11.495933 s

Total Word Count Searches    :        11046
Num Word Count Search Errors :            0
Avg Word Count Search Time   :     0.001047 s
Var Word Count Search Time   :     0.004785 s
Total Word Count Search Time :    11.593082 s

STAT
Avg comparisons per search  -> 55316.806898
List size matches expected? -> Y

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

Total Inserts                :       110462
Num Insert Errors            :            0
Avg Insert Time              :     0.000002 s
Var Insert Time              :     0.000000 s
Total Insert Time            :     0.346291 s

Total Title Searches         :        11046
Num Title Search Errors      :            0
Avg Title Search Time        :     0.000003 s
Var Title Search Time        :     0.000000 s
Total Title Search Time      :     0.048275 s

Total Word Count Searches    :        11046
Num Word Count Search Errors :            0
Avg Word Count Search Time   :     0.000003 s
Var Word Count Search Time   :     0.000000 s
Total Word Count Search Time :     0.043346 s

Press Enter to quit...
```

1 / 2 C Testing part 2 - odd number of books - NB: in the first instance, focus on getting no errors!

Hide Details

My Output on replit and stat function



task2/src/bstdb.c

```
//  + Can you prove that there are no accidental duplicate
document IDs
//    in the tree?
printf("STAT\n");


//Checking if tree is balanced

if (isBalanced(root))
    printf("Tree is balanced\n");
  else
    printf("Tree is not balanced\n");



//Checking if the number of nodes is matching the number of
insertions

if (checkNode(root))
  printf("Number of nodes is as exepected -> %d\n", num_nodes);
else
  printf("Number of nodes is not as exepected\n");



//Average number of comparisons per search
printf("Avg comparisons per search  -> %lf\n",
    (double)g_num_comps / g_num_searches);
}
```

Console Shell

```
Var Word Count Search Time   :     0.217061 s
Total Word Count Search Time :     9.794621 s

STAT
Avg comparisons per search  -> 53457.890932
List size matches expected? -> Y

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

Total Inserts                :       107415
Num Insert Errors            :            0
Avg Insert Time              :     0.000002 s
Var Insert Time              :     0.003499 s
Total Insert Time            :     0.177839 s

Total Title Searches         :        10741
Num Title Search Errors      :            0
Avg Title Search Time        :     0.000002 s
Var Title Search Time        :     0.000039 s
Total Title Search Time      :     0.023361 s

Total Word Count Searches    :        10741
Num Word Count Search Errors :            0
Avg Word Count Search Time   :     0.000001 s
Var Word Count Search Time   :     0.000000 s
Total Word Count Search Time :     0.010029 s

STAT
Tree is balanced
Number of nodes is as exepected -> 107415
Avg comparisons per search  -> 3186.160683
Press Enter to quit...
```

Due consideration given to proper design of the BST to ensure best performance, e.g. what did you do to try and keep the tree balanced?

I have used an AVL tree which is a self-balancing Binary Search Tree where the difference between the heights of the left and right subtrees can't be more than 1 for all nodes

Extra testing to ensure that your BST is performing correctly, e.g how many nodes does it need to visit on average before it responds to a query? You should put the code for this in the stat function.

```
bool isBalanced(struct my_bst* root)
  {
    int lh; /* for height of left subtree */
    int rh; /* for height of right subtree */

   /* If tree is empty then return true */
   if (root == NULL)
      return 1;

   /* Get the height of left and right sub trees */
   lh = height(root->left);
   rh = height(root->right);

   if (abs(lh - rh) <= 1 && isBalanced(root->left) && isBalanced(root->right))
      return 1;

   /* If we reach here then
   tree is not height-balanced */
   return 0;
  }
bool checkNode(struct my_bst *root)
  {
    if (num_nodes == 2^(root->height))
    return 1;
    else
    return 0;
  }

void bstdb_stat(void)
```

```
{



    printf("STAT\n");


    //Checking if tree is balanced

    if (isBalanced(root))
        printf("Tree is balanced\n");
    else
        printf("Tree is not balanced\n");



    //Checking if the number of nodes is matching the number of insertions

    if (checkNode(root))
        printf("Number of nodes is as exepected -> %d\n", num_nodes);
    else
        printf("Number of nodes is not as exepected\n");



    //Average number of comparisons per search
    printf("Avg comparisons per search  -> %lf\n",
           (double)g_num_comps / g_num_searches);
}
```

My output for stat function