Report for ITCS 6114 project 2.

The is implemented in python. There are two major classes. One for AVL and one for Binary search tree. The code automatically takes random input and feeds the same input to both the trees and records the height of the tree. The code takes input from sizes varying from 100 to 100000 over a scale of 1000. and these random input are used to insert into the three.

Data structure used:

Two main classes have been defined, one for AVL and one for Binary search tree. The classes had a node while holds the data value and the address to left and right sub tree. The same type of the data structures are used in both AVL and binary search tree. I have also defines methods called, heights, delete, insert, update heights etc whose functions are self explanatory.
All these classes and methods are individual and there are not dependent so, you can use use it separately.

Executing the code:

Input: The code automatically takes input from random input generation methods and feeds the same input to both AVL and BST. The code automatically executes and finds the height and prints the report.

run it :

```
naveen@ubuntu:~/CCPP/avlt$ python avl.py
```

output:

```
This code automatically takes random inputs of the lengths mentioned in the reslut below
It feeds the same inputs to both Binary search tree and AVL tree.
In the total length insertion is done at probablity of 0.3 and deletion at 0.2 and remaining is spent on searching.
Running...
--Result--
Input data sizes:  [100, 1100, 2100, 3100, 4100, 5100, 6100, 7100, 8100, 9100]
BST heights list:  [11, 29, 44, 54, 61, 80, 85, 104, 112, 118]
AVL heights list:  [7, 11, 12, 13, 13, 13, 14, 14, 14, 14]
```

As you can see in the output the heights generated for different sizes of input are listed and it can deduced that BST develops more height than a AVL for a given same input of same size.

Conclusion:

AVL tree due to their balancing algorithm then to balance their tree stricture and thus reduce their height when compared to BST.