**CS010C Quiz 4: Chapters 7 & 8 Practice Quiz**

*Note: This quiz is much longer than the actual quiz to provide you with various problems for you to practice and understand.*

**Runtime Analysis**

*Please write the runtime using big O notation and justify why the notation is correct.*

1. Height of an AVL Tree / Binary Search Tree / Red Black Tree given a root pointer
2. Travseral (Preorder, InOrder, PostOrder) on a AVL, Binary Search or Red Black Tree
3. Insert n elements into an AVL Tree / Red Black Tree

*For the runtimes regarding hash tables, assume a strong hashing function is applied.*

1. Insert an element into a Hash Table with **closed addressing**
2. Insert an element into a Hash Table with **linear probing**
3. Insert an element into a Hash Table with **quadratic probing**

**Theoretical Analysis**

*For Problems 7 - 12, please write a few sentences answering the prompt. Think carefully about the provided situations / prompts.*

1. What is the difference between a Binary Search Tree and an AVL Tree?
2. What is closed addressing on a hash table? What data structures / methods are used to deal with collisions?
3. What constitutes a Red Black Tree?
4. What is the difference between height and balance factor in an AVL tree?
5. How is the insert operation different between a Binary Search Tree and an AVL Tree?
6. Why is hashing passwords considered good practice compared to storing them as plain text?

**Coding Questions**

*Provided the given class, please code the following functions:*

* Height
* Search
* Largest
* Smallest
* BalanceFactor
* Sum

struct Node {

int val = 0;

Node\* left = nullptr;

Node\* right = nullptr;

};

class AVLTree {  
 private:

Node\* root = nullptr;

int search(Node\*, int) const;

int largest(Node\*) const;

int smallest(Node\*) const;

int sum(Node\*) const;

public:

int height(Node\*) const;

int balanceFactor(Node\*) const;

bool search(int key) const {  
 return search(Node\*, key);

}

int largest() const {

return largest(root);

}

int smallest() const {

return smallest(root);

}

int sum() const {

return sum(root);

}

};

1. Height of Given Node

int height(Node\*) const {

}

1. Search

bool search(Node\* node, int key) const {

}

1. Largest

int largest(Node\* node) const {

}

1. Smallest

int smallest(Node\* node) const {

}

1. Balance Factor of Given Node

int balanceFactor(Node\* node) const {

}

1. Sum

int sum(Node\* node) const {

}

**Visual Diagrams**

*The following questions 19 and 20 require you to visually draw the AVL or Red Black Tree after inserting the specified element(s). DO NOT WRITE ANY CODE. ONLY DRAW/BOX THE RESULTING TREE.*

1. Insert 1, 2, 3, 4, 5 into an AVL Tree

1. Insert 1, 2, 3, 4, 5 into an Red Black Tree

*The following questions 21 - 23 require you to visually draw the Hash Table after inserting the specified element(s). DO NOT WRITE ANY CODE. ONLY DRAW/BOX THE RESULTING HASH TABLE.*

*Assume the following constraints:*

1. 10 Total Buckets
2. Hash Function:

int hash(int val) {  
 return (val \* 2) % 10;

}

1. Insert 1, 2, 3, 4, 11, 22, 33, 44 into a Hash Table with **closed addressing**
2. Insert 1, 2, 3, 4, 11, 22, 33, 44 into a Hash Table with **linear probing**
3. Insert 1, 2, 3, 4, 11, 22, 33, 44 into a Hash Table with **quadratic probing**