Student Name:
Student ID:

CSCE-4133/5133 Algorithms

Midterm Exam

Time: 8:35 a.m. – 9:25 a.m.

Instructions

1. It is an individual exam.
2. You have 50 minutes. You may not leave during the last 10 minutes of the exam.
3. Do NOT open exams until told to. Write your SIDs in the top left corner.
4. If you need to go to the bathroom, bring us your exam, phone, and SID. We will record the time.
5. In the interest of fairness, we want everyone to have access to the same information. To that end, we will not be answering questions about the content.
6. The exam is a closed book.
7. Mark your answers - ON THE EXAM IN THE ANSWER AREAS. We will not grade anything on scratch paper.

Student Name:
Student ID:

Question 1: Please indicate whether each of the following statements is True (T) or False (F) (10 points).

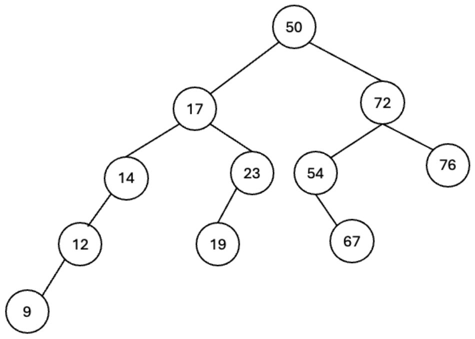
True (A) The complexity of insertion, deletion, and search in AVL Tree is $O(\log n)$.

False (B) In an AVL tree, nodes can have balance factors greater than 1, as long as the difference between their children's heights is less than 2.

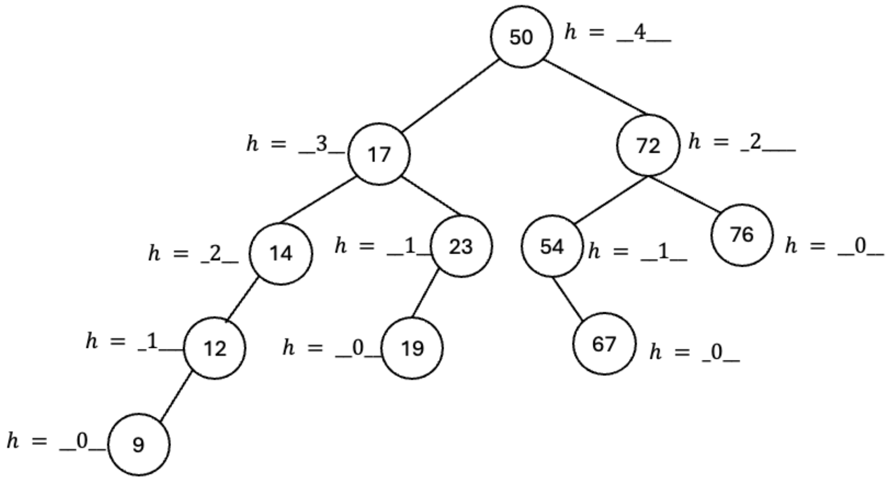
True (C) An AVL tree requires rotations (single or double) to maintain balance after insertions or deletions.

False (D) An AVL tree can become unbalanced if the number of insertions significantly exceeds the number of deletions.

Question 2: Given a binary search tree as follows:



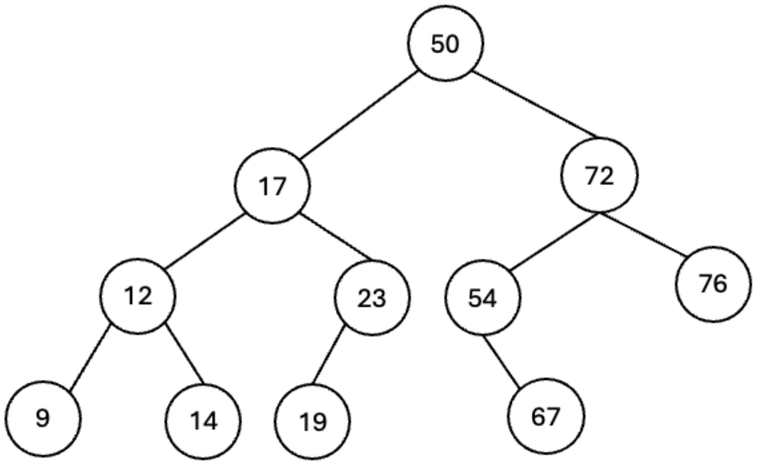
Question 2.1: Calculate the height of each node (10 points).



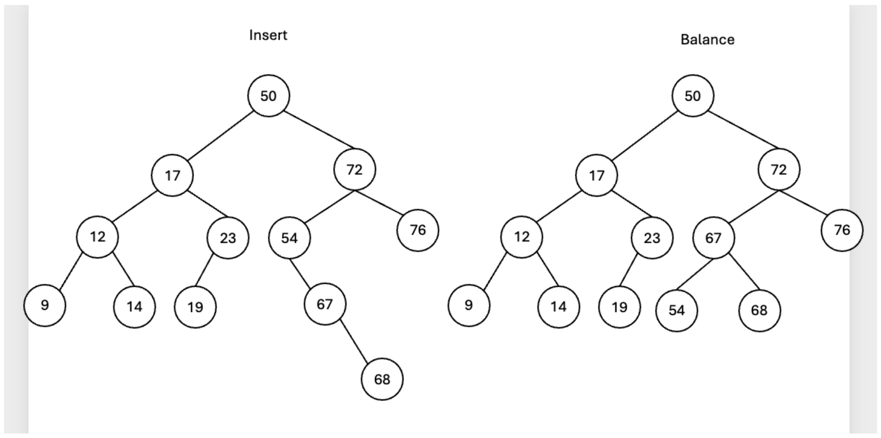
Student Name:
Student ID:

Question 2.2: Is the tree above an AVL tree? Why? If not, kindly perform the necessary adjustments to restore its AVL property. (10 points)

Answer: No. The subtree of 14 do not form an AVL Tree.

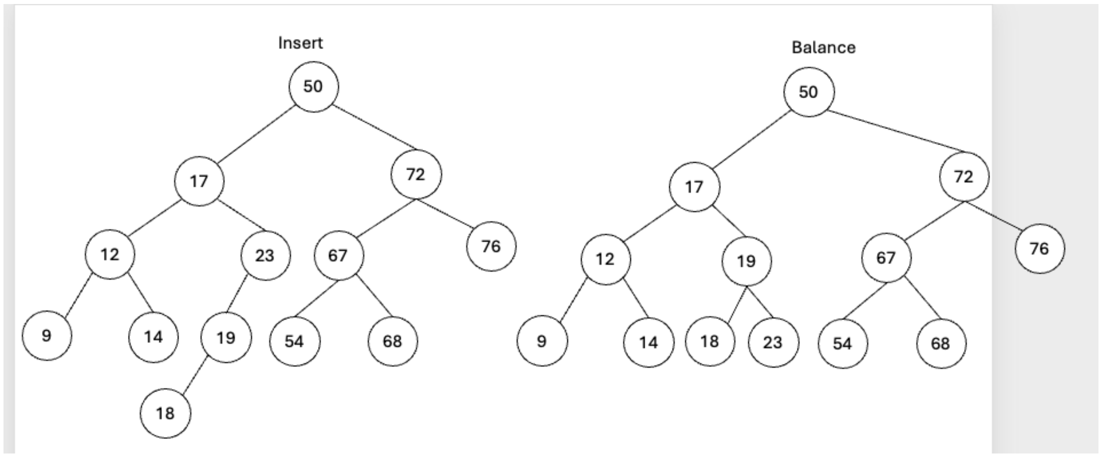


Question 2.3: Insert a value of 68 into the above AVL tree and balance it. If your answer in Question 2.2 is a No, please insert a value of 68 into your AVL Tree after your rebalancing (10 points)

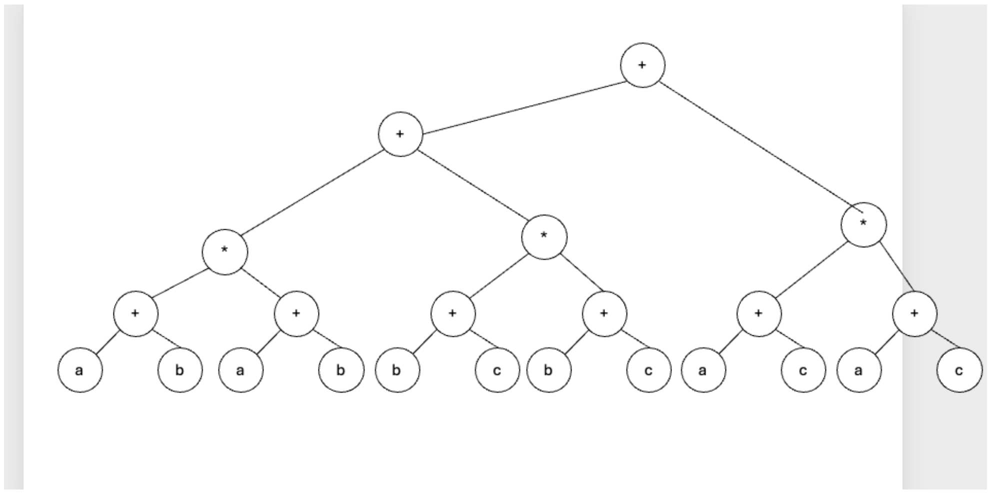


Student Name:
Student ID:

Question 2.4: Insert a value of 18 into the AVL tree obtained in Question 2.3 and balance it (10 points).



Question 3: Build the binary tree with binary operators to represent the mathematical expression $(a+b)^2+(b+c)^2+(a+c)^2$. You are only allowed to use addition (+) and multiplication (*) operands in your binary tree. You are NOT allowed to use power (^) operand in your binary tree (10 points).



Student Name:
Student ID:

Question 4: Given BSTNode and BST structures as follows:

```
struct BSTNode {
    int key;
    int height;
    BSTNode *left;
    BSTNode *right;
};

class BST {45t
    protected:
        BSTNode *root;
    public:
        BST();
        ~BST();
        BSTNode *findNode(int K);
};
```

Question 4.1: Write a function `BSTNode *findNode(int K)` to find the largest node in a Binary Search Tree (BST) whose key is less than a given value K (10 points).

```
BSTNode *BST::findNode(int K) {
    BSTNode *node = this->root;
    int answerNode = NULL;
    while (node) {
        if (node->key >= K)
            node = node->left;
        else {
            answerNode = node; // Possible Solution
            node = node->right;
        }
    }
    return answerNode;
}
```

Question 4.2: Assuming there are n nodes in the BST. What is the time complexity of your implementation? (10 points)

Average-case Time Complexity	Worst-case Time Complexity

Student Name:
Student ID:

O(logn)	O(n)

Question 5: Using the BSTNode structure defined in Question 4, which of the following functions will print the keys of nodes in the BST in the accessing order (10 points)?

(A) `void printNodeA(BSTNode *node)`

(B) `void printNodeB(BSTNode *node)` (This is the answer)

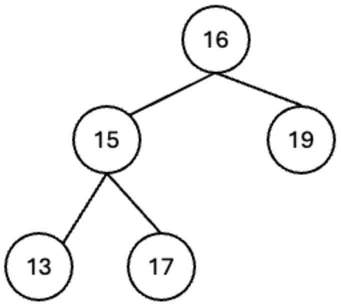
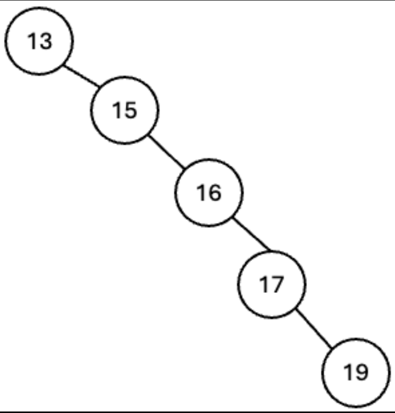
(C) `void printNodeC(BSTNode *node)`

(D) None of above.

(A)	(B)	(C)

`void printNodeA(BSTNode *node) {`
`    if (node == null)`
`        return;`
`    std::cout << node->key << " ";`
`    printNodeA(node->left);`
`    printNodeA(node->right);`
`}`
	`void printNodeB(BSTNode *node) {`
	`    if (node == null)`
	`        return;`
	`    printNodeB(node->left);`
	`    std::cout << node->key << " ";`
	`    printNodeB(node->right);`
	`}`
		`void printNodeC(BSTNode *node) {`
		`    if (node == null)`
		`        return;`
		`    printNodeC(node->left);`
		`    printNodeC(node->right);`
		`    std::cout << node->key << " ";`
		`}`

Question 6: Given the two binary trees as follows:

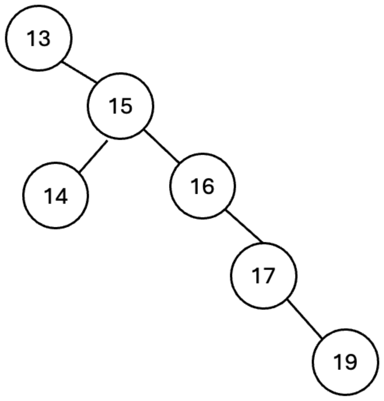
Tree A	Tree B
	

Question 6.1: Which one is the binary search tree? (10 points)

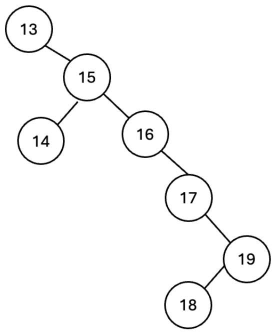
(A) Tree A
(B) Tree B (This is the answer)

Question 6.2: Insert 14 into the binary search tree obtained in Question 6.1 (10 points).

Student Name:
Student ID:

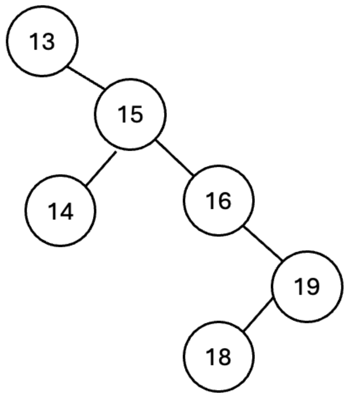


Question 6.3: Insert 18 into the binary search tree obtained in Question 6.2 (10 points).

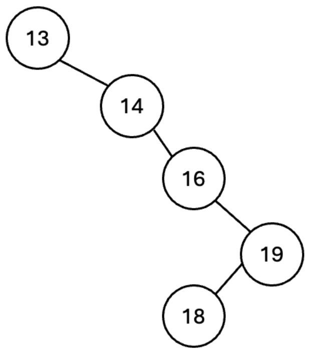


Question 6.4: Delete 17 out of the binary search tree obtained in Question 6.3 (10 points).

Student Name:
Student ID:



Question 6.5: Delete 15 out of the binary search tree obtained in Question 6.4 (10 points).



Question 7: Given the BSTNode and BST structures (bellow), please write a **recursive function** `void releaseBST(BSTNode *node)` to release the memory of the all the nodes in binary search tree (10 points).

```
struct BSTNode {
    int key;
    int height;
    BSTNode *left;
    BSTNode *right;
};

class BST {
    public:
        BSTNode *root;
    public:
        BST();
        ~BST() {
            this->releaseBST(this->root);
        }
        BSTNode *findNode(int key);
        void releaseBST(BSTNode *node);
```

Student Name:
Student ID:

```
};
```

```
void BST::releaseBST(BSTNode *node) {
    if (node == NULL)
        return;
    releaseBST(node->left);
    releaseBST(node->right);
    delete node;
}
```

Question 8 [Graduate Student ONLY]: Given AVLNode and AVL structures (below), please write a function (pseudocode is acceptable) `AVLNode* findSecondMax()` to find the second maximum node. The second maximum in a AVL refers to the second largest key in the tree. You only get a full grade in this question if the complexity of your algorithm is $O(\log n)$ where n is the number of nodes (10 points).

```
struct AVLNode {
    int key;
    int height;
    AVLNode *left;
    AVLNode *right;
};

class AVL {
    protected:
        AVLNode *root;
    public:
        BST();
        ~BST();
        AVLNode* findSecondMax();
};
```

```
AVLNode* AVL::findSecondMax() {
    AVLNode* node = this->root;
    AVLNode* parent = NULL;
    while (node->right) { // Find Maximum Node
        parent = node;
        node = node->right;
    }
    if (node->left != NULL) { // If the max node has the left child, the results will
be the maximum of the left chilren
        node = node->left;
        while (node->right)
            node = node->right;
        return node;
```

Student Name:
Student ID:

```
    } else { // If the max node does not have the left child, the result will the
parent node of the max node
        return parent;
    }
}
```

Question 9: Given an array of n values, kindly provide the time complexity of each sorting algorithm in three scenarios: best case, average case, and worst case (10 points).

Algorithm	Best Case	Average Case	Worst Case
Insertion Sort	$O(N)$	$O(N^2)$	$O(N^2)$
Merge Sort	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
Quick Sort	$O(N \log N)$	$O(N \log N)$	$O(N^2)$
Bubble Sort	$O(N^2)$	$O(N^2)$	$O(N^2)$

Student Name:
Student ID:

Question 10: Given the list of numbers below, run step-by-step the insertion sort algorithm in ascending order (10 points).

9 11 16 23 25 30 37 44 18 13 22

Final Answer: 9, 11, 13, 16, 18, 22, 23, 25, 30, 37, 44 (The student needs to perform step-by-step as the lecture)

Question 11: Given the list of unsorted numbers below, run step-by-step the bubble sort algorithm in ascending order (10 points).

19 26 24 45 17 31

Final Answer: 17 19 24 26 31 45 (The student needs to perform step-by-step as the lecture)

Question 12 [Graduate Students ONLY]: Given the list of unsorted numbers below, run step-by-step the merge sort algorithm in ascending order. The insertion sort algorithm will be called if the list is sorted of size N = 4 or less (10 points).

19 0 50 44 53 27 25 65 35 50 8

Final Answer: 0, 8, 19, 25, 27, 35, 44, 50, 50, 53, 65 (The student needs to perform step-by-step as the lecture)