University of Colombo School of Computing
SCS 1308 - Foundations of Algorithms
Tutorial-12

1.

(a) What is a **DSW Tree**, and why is it used in balancing Binary Search Trees (BST)?
(b) Compare **DSW Tree balancing** with **AVL Tree balancing** in terms of:

- Rotations
- Performance
- Efficiency

2. Complete the missing parts (***1*** and ***2***) in the following **C function** to correctly perform the **right rotations** and create a right-skewed tree.

```c
typedef struct Node {
    int data;
    struct Node *left, *right;
} Node;
// Function to create a new BST node
Node* createNode(int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}
// Function to perform a right rotation
Node* rightRotate(Node* root) {
    if (root == NULL || root->left == NULL) return root;

    Node* newRoot = ___1___;  // Right rotation step
    root->left = newRoot->right;
    newRoot->right = root;

    return newRoot;
}
// Function to create a right vine (backbone)
Node* createRightVine(Node* root) {
    Node* temp = root;
    while (temp != NULL) {
        if (temp->left != NULL) {
            temp = ___2___;  // Apply right rotation on the left child
        } else {
            temp = temp->right;
        }
    }
    return root;
}
```

3. The following **C function** contains errors in the **left rotation implementation**. **Identify and fix the errors**.

```c
Node* leftRotate(Node* root) {
    if (root == NULL || root->right == NULL) return root;

    Node* newRoot = root->right;
    root->right = newRoot->left;
    newRoot->left = root;

    return root;  // ERROR: Incorrect return value
}
```

4. Write a **complete C program** to:

- Construct a **BST** from user input.
- **Convert it into a right vine**.
- **Apply left rotations** to balance the tree.
- **Print the tree** before and after balancing.

Hint: Use **logarithm-based calculations** to determine the number of rotations.

5. Consider a **Binary Search Tree (BST)** with **15 nodes**.
   (a) How many **right rotations** are needed to create the right vine?
   (b) How many **left rotations** are required to balance the tree?

6. Modify the **recursive DSW tree balancing algorithm** to an **iterative version** using **loops** instead of recursion.
Ensure that the program:

- **Uses an iterative loop** to create the right vine.
- **Uses an iterative loop** to perform left rotations.
- **Accepts user input** for building the BST.