Lab #10
Spring 2025

Requirements

In this lab, you'll expand your understanding of Binary Search Trees by implementing additional features and operations.

```
typedef struct BSTNode {
    int data;
    struct BSTNode* left;
    struct BSTNode* right;
} BSTNode;
typedef struct BST {
    BSTNode* root;
    int size;
} BST;
```

1 initBST

```
// O(1)
BST* initBST ();
```

Initializes and returns a pointer to a BST struct on success, or NULL on failure.

2 insertBST

```
// O(log n)
int insertBST (BST* tree, int data);
```

Inserts a **new integer** into the BST. Return 0 on success, 1 on failure.

3 getHeightBST

```
//  O(n)
int getHeightBST (BST *tree);
```

Returns the height of the BST. Height is defined as the number of edges in the longest path from the root to a leaf.

4 getMinimum

```
// O(log n)
int getMinimum (BST *tree);
```

This function returns the minimum value stored in the BST. It does so by traversing the leftmost path from the root node. Return -1 if the tree is NULL or empty.

5 getMaximum

```
// O(log n)
int getMaximum (BST *tree);
```

This function returns the maximum value stored in the BST. It traverses the rightmost path from the root node. Return -1 if the tree is NULL or empty.

6 isValidBST

```
// O(n)
int isValidBST (BST *tree);
```

This function verifies that the current structure of the binary tree satisfies the Binary Search Tree (BST) property, where:

- Every node in the left subtree contains only values less than the node's value
- Every node in the right subtree contains only values greater than the node's value

The function should perform a recursive traversal of the tree, verifying that all nodes lie within valid value bounds. Return 1 if valid, 0 if any rule is violated.

Hint: Use a helper function with min and max bounds to validate each node recursively.

7 freeBST

```
// O(n)
void freeBST (BST *tree);
```

Frees all memory used by the BST.

Rubric

- (2 pts) initBST() correctly initializes an empty tree
- (3 pts) insertBST() correctly inserts nodes, handles duplicates
- (4 pts) getHeightBST() returns correct height (edges, not nodes)
- (4 pts) getMinimum() and getMaximum() return correct values, handle empty tree case
- (4 pts) isValidBST() correctly checks BST properties, uses min/max logic
- (3 pts) freeBST() and helpers properly deallocate all memory