Pre-Lab Write Up

Name: Abby, Aliza, and Ro

Lab: #6 Binary Search Tree Dictionary

1. Describe in English what this program is supposed to do (not how it does it). **This should be able to be your class comment at the top of your program (you may copy and paste this into your program later)**:

```
This program creates an address book of people and their contact information that
can be sorted 3 ways (inorder, preorder, postorder).  A user can add, delete, and
modify a record providing they know the record key. The number of records (nodes)
can be counted.
```

2. & 3. & 4.

Public

- `DatabaseMethods()` - Constructor to initialize the root.

Parameters: None

Returns: None (Constructor)

Exceptions: None

Description: Initializes the `DatabaseMethods` object with `root` set to `null`, creating an empty binary search tree.

- `getRoot()` - Returns the root of the tree.

Parameters: None

Returns: `DatabaseNode` - The root of the binary search tree.

Exceptions: None

Descriptions: Simply returns the current root of the tree, allowing access to the tree's topmost node.

- `addNode(DatabaseNode newNode)` - Adds a new node to the binary search tree.

Parameters:

- `newNode` (`DatabaseNode`) - The node to be added to the tree.

Returns: None

Exceptions: None (may print an error message for duplicate nodes)

Description: Checks if the tree is empty; if so, sets the `newNode` as the root. Otherwise, traverses the tree starting from the root to find the appropriate position for the new node based on its ID, inserting it in either the left or right subtree.

- `generateID()` - Generates a unique random ID.

Parameters: None

Returns: `int` - A unique random ID.

Exceptions: None

Description: Uses a random number generator to create a unique integer ID. It ensures uniqueness by checking the `listIDs` array and regenerating IDs if duplicates are found.

- `deleteNode()` - Prompts the user to delete a node by ID.

Parameters: None

Returns: None

Exceptions: `InputMismatchException` (from `scanner.nextInt()` if the input is not an integer)

Description: Prompts the user for an ID and calls a private helper method to remove the node with the specified ID. It adjusts the tree's structure depending on whether the node has no children, one child, or two children. Recursively traverses the tree to find the node with the specified ID. Once found:

- Deletes it directly if it's a leaf node.
- Replaces it with its only child if it has one.
- Finds its successor (minimum node in the right subtree) to replace it if it has two children.

- `modifyNode()` - Modifies a node's data based on user input.

Parameters: None

Returns: None

Exceptions: `InputMismatchException` (from `scan.nextInt()` if the input is not an integer)

Description: Prompts the user for an ID to locate the corresponding node. Provides options for modifying specific attributes of the node. Deletes the node, creates a new one with the updated data, and re-adds it to the tree.

- `lookupNode()` - Prompts the user to look up a node by ID and prints in the specified order.

Parameters: None

Returns: None

Exceptions: `InputMismatchException` (from `scanner.nextInt()` if the input is not an integer)

Description: Prompts the user for an ID and searches for the corresponding node. If found, allows the user to print the subtree rooted at the found node in preorder, inorder, or postorder traversal.

- `printPreorder(DatabaseNode root)` - Prints nodes in preorder traversal using iteration and a stack.

Parameters:

- `root` (`DatabaseNode`) - The root node of the subtree to print in preorder.

Returns: None

Exceptions: None

Description: Uses a stack to perform a non-recursive preorder traversal of the tree. Visits the root, then left and right subtrees in sequence.

- `printInOrder(DatabaseNode node)` - Prints nodes in inorder traversal recursively.

Parameters:

- `node` (`DatabaseNode`) - The root node of the subtree to print in inorder.

Returns: None

Exceptions: None

Description: Implements a recursive inorder traversal. Visits the left subtree, root, and then the right subtree.

- `printPostOrder()` - Prints nodes in postorder traversal using iteration and a stack.

Parameters: None

Returns: None

Exceptions: None

Description: Uses a stack and flags to perform a non-recursive postorder traversal. Traverses left and right subtrees first, then visits the root.

- `createNode()` - Creates a new node from user input.

Parameters: None

Returns: `DatabaseNode` - A new node created from user input.

Exceptions: `InputMismatchException` (if user input for `zip` is not an integer)

Description: Collects user input for various fields like name, address, and phone number to create a new `DatabaseNode`. Uses the `generateID` method to assign a unique ID to the node.

- `countRecords(DatabaseNode node)` - Counts and returns the number of records in the tree recursively.

Parameters: `node` (`DatabaseNode`) - The root node of the subtree to count records.

Returns: `int` - The total number of records (nodes) in the subtree.

Exceptions: None

Description: Recursively traverses the tree to count all nodes. Adds 1 for the current node and recursively counts nodes in the left and right subtrees.

Private

- `findMin(DatabaseNode node)` - Finds the node with the minimum ID in the tree.

Parameters:

- `node` (`DatabaseNode`) - The root of the subtree to find the minimum node.

Returns: `DatabaseNode` - The node with the smallest ID in the subtree.

Exceptions: None

Description: Traverses the left children of the given node until the smallest value (leftmost node) is found and returns it.

- `search(int idNum, DatabaseNode node)` - Searches for a node by ID in the tree

Parameters:

- `idNum` (`int`) - The ID to search for.
- `node` (`DatabaseNode`) - The root of the subtree to search within.

Returns: `DatabaseNode` - The node with the matching ID, or `null` if not found.

Exceptions: None

Description: Recursively traverses the tree to find a node with the matching ID. Moves to the left or right subtree depending on whether the ID is smaller or larger than the current node's ID.

3. What questions do you still have about this lab after reading through the specification and completing the pre-lab?

These records have many pieces of data and did not come with a search method for users to find a record they may need to modify or delete once a record was added. A method was added to print the nodes so that a user can then see the key of the node they wanted to modify.