CISC220 Homework 3

Lab portion due October 19th at midnight on Sakai. The paper copy is due Tuesday, October 20th in lab.

Each part is marked as **[PAIR]** or **[INDIVIDUAL]** and must be completed accordingly.

Do not use any classes from any template library; the point of these exercises is for you to code from scratch. All work must be typed, including individual short answer questions.

Academic Honesty Reminder

For parts that are marked as **[PAIR]** you may work with one other student. Both students must submit separate copies of the assignment and include the names of both members of the group.

For parts that are marked as **[INDIVIDUAL]** you may discuss general algorithms with other people, but you may not share any of your code in **any** form. You may not help other people debug their programs, except in the very limited way described on the class website. You may not use any code written by others, whether they are students or not. The TA and the instructor are available for office hours, and by appointment if you can't make hours because of a class conflict.

Assignment

1. **[PAIR] 50 pts.** Consider the ordered BinaryTree implementation available on the course website under homework 3. This implementation is structured similarly to our LinkedList; the BinaryTree class contains a pointer to the root BinaryTreeNode. Each BinaryTreeNode has a data value, a pointer to its left child and a pointer to its right child. Your task is to implement the following functions on the BinaryTreeNode in the BinaryTree.h file but are not yet implemented in BinaryTree.cpp:

```
/**
 * Returns true if the given integer key is a member of the
 * Binary Tree rooted at this Node.
 */
bool member(int key);

/**
 * Removes the node that matches the given integer key from the
 * Binary Tree rooted at this Node.
 *
 * Returns the removed BinaryTreeNode, with its left child set
 * to the new root of this subtree.
 */
BinaryTreeNode* removeKey(int key);
```

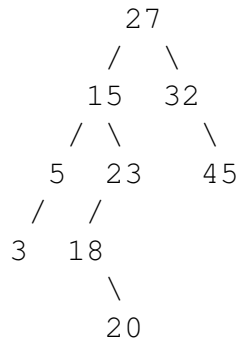
```
/**
 * Inserts the given NodeData into the subtree rooted at
 * this BinaryTreeNode.
 */
void insert(NodeData *newData);

/**
 * Returns the first (smallest value key) NodeData of the subtree
 * rooted at this BinaryTreeNode.
 */
NodeData* first();

/**
 * Returns the size of the subtree rooted at this node.
 */
int size();
```

- Create and edit BinaryTree.cpp to implement the above BinaryTreeNode functions. You should not have to change anything in the BinaryTree class itself.
- Modify BinaryTreeTest.cpp to include additional tests and corresponding output for each of the newly implemented functions.
- Based on your code, calculate the order of the running time for each function and put this in your comments for the function (for example, as size should have an O(n) running time).
- Submit your implemented code and a script file that uses BinaryTreeTest to test each of your new functions.

2. **[*PAIR 0/50 pts.]** Using your code from Part 1, implement a pre-order traversal of the binary tree. Your traversal function should display the NodeData-> key for each node, separated by a comma (it is okay if the last element has a comma following). For example, a pre-order traversal display for this tree:



Would result in the output:

```
27,15,5,3,23,18,20,32,45
```

Your solution must clean up ALL memory allocated from the heap before exiting the main function. You must also document every function in your code and use descriptive variable names.

Submission of completed Homework 3 is due October 19th at midnight. You should submit the following to Sakai and printed copies to your TA:

1. Code for Part 1, including Makefile.
2. Script output showing execution of Code for Part 1.
3. Code for Part 2, including Makefile.
4. Script output showing execution of Code for Part 2.