Pre-Programming Approach

4/10/23:

1. Review assignment instructions and determine concepts to study/review:
    a. Pretty confident in the inorder traversal concept, printing each step should be simple. Reverse order should be simple as long as it follows one of the traversal concepts from the text (if not, could simply run the inorder traversal again and store to a list and print the reverse order.)
    b. Doing step a without recursion might be a challenge, will need to review ADT examples in the text to see how this might be accomplished.
    c. I have worked with threading on past work projects in the past, just need to research methods on how to implement in C++
    d. BSTNode.h:
        i. Figuring out whether a node pointer is a thread or regular pointer seems simplistic in concept. Just find a tangible quality that these two items do not share and test for it.
        ii. Determine how the threads will run, as we cannot add additional pointers
        iii. Add getters and setters for i (simple).
    e. BST.h:
        i. Rewriting inserthelp() to accept a pointer or thread will be challenging, as most of my programming experience has been with single typed parameters. The last project (bag/dictionary) was my first real exposure to templates, I feel I may need to review them and associated concepts.
        ii. Rewriting printhelp() to work with threads, difficulty will largely depend on how difficult threading in C++ is, and what steps to accommodate it will be needed.
        iii. printInorder() without recursion is a concept I will need to review, as recursion was my first plan in attempting these functions. Will need to review book concepts, as I remember a blurb about the high costs of recursion, perhaps my plan lies in there?
        iv. printReverse() should simply utilize the same logic as printInorder, but store to a list and read the elements backwards.
        v. I am able to add private methods to make the above methods easier, the usefulness of this will be determined later, but it's a good fallback to be aware of.
        vi. Destructor will not work with threaded trees
        vii. Functions that are commented out by instructor do not need to be implemented, only above specified functions are required.
    f. Looks like I have to write my own .cpp driver, this shouldn't be an issue
        i. Write simple print statements for each function so I know they are being implemented and called.
        ii. Comments on everything
    g. BSTNode does not seem like a difficult part of the project once threading is understood, the true challenge is the BST.h file:

        i. When writing insertHelp(), I must consider node states, and how each state affects the insert operation. (I assume this relies on the isPointer() variables)
2. After reviewing the assignment instructions, this is my logical flow for inserthelp():
    a. For each node in BST:
        Check the state of the node (does node have value and two children?)
        If the node is full:
            Continue to next node
        Else:
            Determine which child to insert kvpair
            Call insert()
            Break
3. Review project 5.2 from textbook:
    a. Uses pointers that would be null and makes them threads in a BST to avoid wasted overhead
    b. Uses the node implementation of figure 5.7, but each node has two additional bit fields to determine if left child and right child are regular pointers to child nodes or threads.
    c. If a left child pointer would be NULL in a normal binary tree, then it instead stores a pointer to the inorder predecessor of that node (the node that would be printed immediately before the current node)
    d. If a right child pointer would be NULL in a normal binary tree, then it instead stores a pointer to the inorder successor of that node (the node that would be printed immediately after the current node)
    e. Steps c and d allow for traversals without recursion!
4. Tomorrow's plan of attack:
    a. Write .cpp file and make sure everything runs correctly
    b. Research more about what makes threads able to do traversals without recursion.
    c. Implement methods using figure 5.7, adding bit fields for the lc and rc.

Programming Approach

4/11/2023:

1. Downloaded project files from canvas
2. Put comments where needed for readability
3. Write .cpp file for project:
    a. Make calls to the printhelp, printorder, and printreverse function
        i. These functions will be simply printing “This is the _____ function, call was successful.
        ii. Seems like the inorder/reverse functions are completely omitted. I was able to insert and print the binary tree, but will have to take additional measures before I can test the other functions.

4/12/2023:

1. Added boolean data variables to BSTNode.h. Will try to change to bit fields if I have time at the end of the project.
2. Started modifying the setleft and setright functions. While I’m not entirely sure how to direct these pointers yet, I used a simple if/else to check whether the child is null (based on the bool isLeaf() function) to set its thread variable.
3. Wrote getThreadStatusLeft and Right to return the boolean variable. I believe these will be crucial in determining where child pointers are mapped.
4. Started rewriting inserthelp()
    a. Gotta check if recursion is ok for this, I would assume so since the recursion limitations are only referring to the traversals in the instructions.
    b. If a tree is empty when the first insertion happens for the root node, how does this affect the threads? Does the left simply point to root and the right remain null?
    c. I went through the current inserthelp() function and it all makes sense. Now I will have to change the thread status as the left or right child is created
    d. After a few hours I took a break, running into an infinite loop where it only prints out the last insert statement, assigning it to a right child each time

4/13/2023

1. Got the infinite loop to stop by changing the child initialization values to NULL.
    a. Now inserthelp is omitting some values
    b. Fixed 1a by adding a logical flow element that I overlooked (checking thread status)
2. Plan is now to assign thread pointers, I think I’ll do this within the BSTNode.h file
    a. Not exactly sure if pointers are allowed to assign this
    b. Where should the threads be assigned? Hard to figure this out without pointers being added
3. *Ask Mr. Christian:*

- *I want to make sure I understand threading in the sense we are being asked to implement it Am I correct when I assume that the threads need to literally point (using*

the lc/rc pointer) to a node depending on whether it's a right child or left child? Or are the boolean flags replacing this need (Ex: if (this is a thread), check if right or left child and find/print the appropriate location, else: print this and continue inorder traversal) if I am not required to literally assign the child pointers, please disregard the next question

- *I did have a difficult time with the inserthelp(), but I feel like I'm struggling more with assigning the children of nodes to thread to the correct location. Should I be assigning these threads within BSTNode.h? Or passing them as parameters in the inserthelp() node generation?*
- *In order to find the predecessor and successor nodes for an inorder traversal, I plan to make a helper function(s) to find these. I saw in the assignment instructions that we cannot add pointers to the BSTNode.h implementation. Can I add pointers to the BST.h file or is that restricted as well? I'm not sure of a method of finding these nodes without allocating some form of memory*

4/16/2023:

- For now, I will assume physical assignment of child pointers is not important, and that the "threads" simply check if a child has the thread flagged or not
- Will assume that pointers can be added to files that aren't the BSTNode.h file.

1. Started working on the inorder traversal method.
    a. I have to avoid using recursion, so I will try to use a while loop to follow the right sided threads
    b. Will also create a help function to determine the next node to visit (Ex: if rc is a thread, follow thread, else continue inorder traversal mapping.
        i. Might have to do a separate helper to find predecessors
    c. I can see why the instructions said insertHelp method needs to be able to deal with threads, getting a ton of exceptions due to null pointers
        i. Will put logic in to deal with this and/or potentially modify the inserthelp() method
    d. Modified printhelp to ensure no nullptrs are dereferenced
    e. Got printInorder() to work, with the SuccessorFind helper method.
        i. Assuming the printReverse and PredecessorFind methods will be simple, as the right/left thread checks will just need swapped.
2. Started working on printReverse
    a. Just as stated in 1d-i, this was almost a copy/paste clone of the printInorder and SuccessorFind methods, but with just following the opposite threads.
    b. Had a couple exceptions raised, but they were solved similar to the previous ones.

4/17/2023:

- With today being the due date, I will omit the attempt to incorporate bit fields

- Need to add english header outputs to .cpp file so it matches the assignment instructions example
- Need to work on screenshots
- Put comments in code where needed

1. Added console outputs for code readability to .cpp file
    a. Almost matches the example from instructions perfectly, but I added more spacing for aesthetics.
2. Comments have been added to member functions that were extensively modified
3. Screenshots and integrity statements added.