LAB EXERCISE

Inorder

Background:

Any recursive algorithm can be reduced to a linear sequence of events. It will be longer and more difficult to follow, but recursive solutions can be rewritten as iterative solutions if a stack is available for use. In this lab exercise, you will implement a non-recursive `inorder` method now that we have a `stack` class to support stack operations. After completing the lab, you should have a greater appreciation for recursion and what it will accomplish for you.

You will work with the same binary tree code as implemented in Lessons 33-35. A non-recursive `inorder` method is summarized in pseudocode form below.

```
void inorder (TreeNode root)
{
  declare a stack of TreeNode, initialized as empty
  declare temp as a TreeNode

  start temp = root

  do
  {
    while moving temp as far left as possible,
      push tree references onto the stack

    if the stack is not empty
      reposition temp by popping the stack

    print the contents of tempgetValue()
    move temp one node to the right
  }
  while (the stack is not empty) or (temp != null)
}
```

Assignment:

1. Starting with an old binary tree lab, keep the code needed to read a data file (*file20.txt*) and build the binary tree.

2. Implement the `Stack` interface using either the `ArrayStack` class or the `ListStack` class described in the notes.

3. Solve the code for the non-recursive `inorder` method. Use (*file20.txt*) to test your program.

Instructions:

1. Display your source code and a run output. The run output should consist of the `inorder` output of the binary tree. Call your instructor to your workspace for scoring.