**Detailed Algorithm and Program Design:**

**Error Checking Class**

--create a bool function to check spacing of file

Check the first array of the string input from file, if the second element isn’t a space return false, else return true

**BSTree Class**

**Create Error Checking Function in Bst Class**

--create a new class in charge of making the bst

In the Header file:

Create a Node Struct:

--Make a struct for the node, the points to the left and right child

--Make a constructer for the node that takes data

--Set value of current node equal to the data

--make left and right nodes null pointers

--make a height int to check the height of the tree

Make a Node Constructer to set the values

Make a right rotating node to rotate a node to the right

Make a left rotating node to rotate a node to the left

Make a Balance number to keep track of each nodes balance

Set the current right and left node and return there difference in height, if right is a null pointer return 1, and if left is a null pointer return 0.

In the BST Cpp file:

Make a height and operation integer and set both to zero

Make a function to get maximum depth

Check if root is null

If not, traverse the tree from root to the left and then right, increment depth each time then return the maximum depth when null is reached

Make a Boolean function to see if tree is empty

If the root is null return true

If not return false

Make a void function to insert a node into the tree that takes a root and int

--If the root is null print the tree has no root to insert! Attempting to create tree root, then create a root and set it to the given value

If root is not empty:

--Compare the root and int, if int is bigger go left of root, else go right and repeat the comparing until we reach a null node, then set the value there

Make a function to print the tree

Check if tree is empty, and If it is tell user , if not, make a prefix for printing the tree that takes a bool to check if the node is on the left, a node, and a prefix, then if the node is on the left print the corresponding graph, if not, print the corresponding graph for nodes on the right, call print tree recursively and check if it is left in the argument, send in the node on the left, and a true, call another print tree but send the node on the right and false

Make a function to start the print tree function

Just call print tree , and send in a empty string, the root and a false

Make a function to search the tree that takes a node and a key int

If the node is false tell user and return a nullptr

Otherwise make a current node that points to the

**Main**

a) open file and check if it is open

i. if the file did not open, display error message and exit

b) get expression from file until the end of the file line is reached

c) call every error checking function from error checking class

i. if any of the function equal false check which

a. if the spacing function equals false, print : “Error: invalid spacing”

b. if the capitalization function equals false, print “Error: inaccurate capitalization”

c. if the parentheses function equals false, print “Error: mismatching parentheses”

d. if the formatting function equals false, print “Error: inaccurate formatting”

e. exit the program

ii. if all functions equal true

1. call the first function from the node class using a node object
2. call the function to build a tree from the binary tree class using a binary tree object. This function will call all the other functions in the class.
3. call the height function from the Binary tree class and set it equal to an integer variable
4. call the printing tree function and pass in the top node and the height.
5. call the numbers class and pass in the postfix expression to the first function.

Create a function to count the performed operation

Create a function to count height