Project 3 Pseudocode

**Insert**

* This is a recursive function that takes in a root node and a node to insert and returns nothing.
* The base case is when the child of the current node is a nullptr. At this point add the new node to that nullptr location, link the pointer between child and parent, change the child pointers to nullptr, and update the number of nodes in the tree.
* While not in the base case, decide whether to go right or left based on alphabetizing by the title of the movie.

**Search**

* This is a recursive function that takes in a pointer to root and pointer to a node to look for and returns a pointer to the found node
* If the root is null or the root equals the node you’re looking for
  + Return root
* If the title of the node you’re looking for is higher than the current node
  + return the function by passing in the left child of the current node
* If the title of the node you’re looking for is lower than the current node
  + Return the function by passing in the right child of the current node

**Delete**

* This is a recursive function that takes in the root node and the node to delete, and returns the depth of the node deleted (int)
* If the root is null
  + Return –1 as the depth
* If the title of the node to delete is less than the title of the current node
  + Set depth equal to the function, which is called by passing in the node to the left of the current node and the node to be deleted
  + Return the depth
* If the title of the node to delete is greater than the title of the current node
  + Set depth equal to the function, which is called by passing in the node to the right of the current node and the node to be deleted
  + Return the depth
* If the left and right nodes of the current node are not null
  + Set a temp variable equal to the min of the right and left
  + Run delete with the tmp->right and the min of right and left of the current node
  + Return 0
* If the current node->left is nullptr
  + Set the root equal to the root->right
* Else
  + Set the root equal to the root->left
* Delete the temp version of the root
* Return 0

**main.cpp**

* Function: parse to create BST, which takes in a binary search tree to parse into and returns nothing
  + Open up the file inventory.dat
  + While loop parsing a line at a time
    - Create a new node, giving it the title and number available from the data file
    - Insert the new node into the BST
  + Close the file inventory.dat
* Function: parse to go through transactions which takes in the BST and returns nothing
  + Open up the file transaction.log
  + While loop parsing a line at a time
    - If add
      * Validate format
      * Create a new node, giving it the title and number available from the data file
      * Insert the new node into the BST
    - If remove
      * Validate format
      * Search to make sure the node is actually in the BST
      * Run delete function for that node
    - If rent
      * Validate title
      * Run rent function
    - If return
      * Validate title
      * Run return function
    - Else
      * Write error to error log
* Function: rent, takes in a title to rent and returns nothing
  + Ensure the title exists in the system with search
  + Traverse the BST (as previously described in search function) until the proper title is found
  + Increase the number of rented and decrease the number of available DVDs
* Function: return, takes in a title and returns nothing
  + Ensure the title exists in the system with search
  + Traverse the BST (as previously described in search function) until the proper title is found
  + Increase the number of available and decrease the number of rented DVDs
* Function: print, takes in the BST and the file object and returns nothing
  + This is recursive
  + If the node is null, return
  + Print(left)
  + Print out the information for the node
  + Print(right)
* The main function creates a BST and runs both parse functions
* Then print the report to the file with the print function, first creating a file out object to pass in