**Project 3 Pseudocode**

**Main**

**Main**

* Open inventory.dat
* createTree tree
* open transaction.log
* readTransaction

**createTree**

* **Parameters:** ifstream input
* **Return:** BinaryTree tree
* Create tree
* While input is not at the end of the file
  + Store title
  + Store available
  + Store rented
  + Create Node (title, available, rented)
  + Insert node into tree
* Return tree

**readTransaction**

* **Parameters:** ifstream input, Tree tree
* **Return:** void
* While input is not at the end of the file
  + read line
  + if line does not follow format guidelines
    - print to error.log
    - continue loop
  + if line starts with add
    - add
  + if line starts with remove
    - remove
  + if line starts with rent
    - rent
  + if line starts with return
    - return

**add**

* **Parameters:** string line, Tree tree
* **Return:** void
* Store title
* Store number to add
* Search tree for node with title
  + if node does not exist
    - insert new node with title
* Increase node available amount by number to add

**remove**

* **Parameters:** string line**,** Tree tree
* **Return:** void
* Store title
* store number to remove
* search tree for node with title
* decrease node available amount by number to remove
* if node available amount <= 0 and node rent amount <= 0
  + delete node from tree

**rent**

* **Parameters:** string line**,** Tree tree
* **Return:** void
* Store title
* Search tree for node with title
* Decrease node available amount by one
* Increase node rent amount by one

**Return**

* **Parameters:** string line**,** Tree tree
* **Return:** void
* Store title
* Search tree for node with title
* Decrease node rent amount by one
* Increase node available amount by one

**Binary Search Tree**

**Insert**

* **Parameters:** Node node, string title
* **Return:** node
* if node title is the same as title
  + increment node available
* If node is null
  + Return New node with title
* if title < node title
  + node left = insert(node left, title)
* if title > node title
  + node right = insert(node right, title)
* return node

**Search**

* **Parameters:** Node node, string title
* **Return:** node
* If node is null or node title equals title
  + Return node
* If node title < title
  + Return search(node right, title)
* If node title > title
  + Return search(node left, title)

**Delete**

* **Parameters:** Node node, string title
* **Return:** node
* If node is null
  + Return node
* If title < node title
  + Node left = delete(node left, title)
* else if title > node title
  + Node right = delete(node right, title)
* Else
  + if node left is null
    - store node right
    - delete node
    - return node right
  + else if node right is null
    - store node left
    - delete node
    - return node left
  + store min value of tree node right
  + set node info to stored node info
  + set node right to delete(node right, title)
* return node