Password Authentication System (Pseudo-code):

Creating an account:

* Have the user create a username and password, each consisting of letters and maybe numbers/special characters
* When the user clicks the submit button, try to determine if the user’s inputted account name is already in the database
* Start with the first element of the tree, containing the name of the first account created for the database
* Compare the usernames character by character
  + If the user’s inputted account name equals the current node’s account name, return an error message (the account name is already taken)
  + Else, if the user’s inputted account name < current node’s username, go to left child
  + Else, the user’s inputted account name > current node’s username, so go to right child
* The process of finding the user’s account name in the database will continue through the child nodes until either the algorithm reaches a node with a matching username, or the algorithm tries to find the username in a node that does not exist (the node is null, nullptr, or something like that)
  + If the algorithm eventually reaches a point where it cannot find the given username (it reaches a nonexistent node), then the username is available
    - Create a new node with the user’s given account name and password, before linking that node with the last node that the algorithm passed before hitting null

Login

* Have the user input a password, consisting of letters and maybe numbers/special characters
* When the user clicks the submit button, try to find the user’s inputted password in the database of stored accounts and passwords
  + The case should matter
  + Preferably, the database of stored accounts and passwords should be in the form of a tree, since searching in a tree is more efficient than iterating through a large list
    - Each node of the tree contains an account name (a char array) and password (also a char array), along with two pointers to the left and right child nodes respectively
  + If the user has left the username or password fields blank, simply return an error message (please input an account name and password)
* Start with the first element of the tree, containing the name of the first account created for the database
* Compare the usernames character by character
  + If the user’s inputted account name < current node’s username, go to left child
  + Else, if the user’s inputted account name > current node’s username, so go to right child
  + Else, the user’s inputted account name matches the current node’s username, so determine if the inputted password matches the current node’s password
    - If the two passwords match, login the user
    - Otherwise, print an error message (Incorrect password)
  + The process of finding the user’s account name in the database will continue through the child nodes until either the algorithm reaches a node with a matching username, or the algorithm tries to find the username in a node that does not exist (the node is null, nullptr, or something like that)
    - In the instance where the account name does not exist, return an error message (the username does not exist)