Read a list of names from a file. Insert the names into a binary search tree that is a little different from what we discussed in class. For this tree, the left side will contain the larger values and the right side will contain the smaller values. In essence, it is the exact opposite of a normal binary search tree.

Our tree will also be able to store duplicate names. Aside from a left and a right pointer for each node, we will have another pointer variable that will point to a linked list. Each node in the linked list will contain the additional duplicate name.

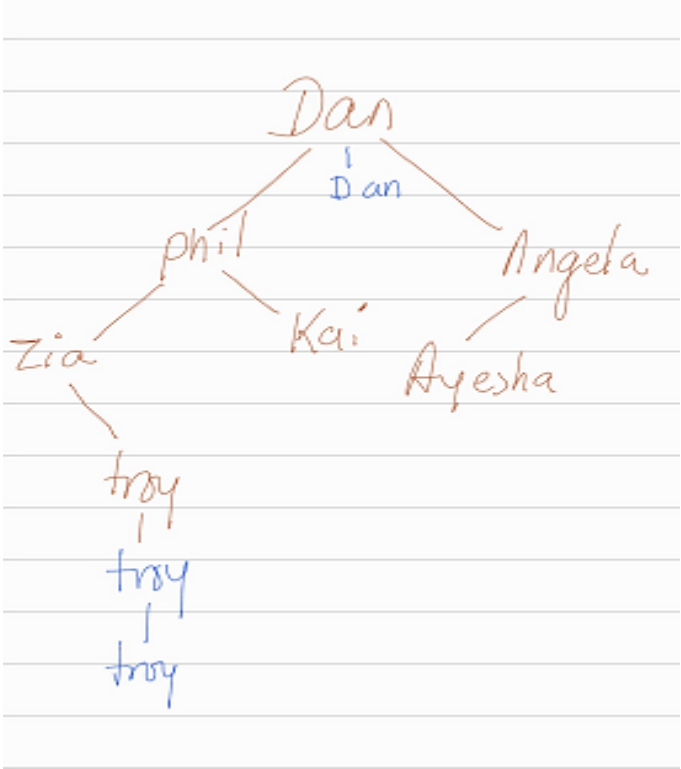
Once the nodes have been inserted, give the user a chance to search for a name. Display your search path as you are searching through the tree and indicate if the name was found. If the name is found and there are duplicates, indicate how many duplicates of the name exist in the tree.

Here is an example:

Given the following names in the file:

Dan, Phil, Angela, Ayesha, Kai, Zia, Troy, Troy, Troy, Dan

Your tree would look like the following:



The Names in blue are in the linked list. For the search, if the user is looking for Troy, you would display the search path as such: Dan, Phil, Zia, Troy. You would also indicate that there are 3 iterations of Troy.

Hints:

When comparing strings, you would use the strcmp function which does a case sensitive compare. You can also do strcmpi which doesn't care about the case. You can assume that we don't care about the case, which means that Dan and dan are both the same names.

Also, the names will have to go into a char array, such as char name[30]. You can assume that names will not exceed 30 characters.

When copying strings from one array to another, you have to use the strcpy function.

```
Here is what the structure should look like
struct node {
      char name[30];
      struct node* next;
};

struct BinaryTree {
    char name[30];
    node *names;
    BinaryTree *left;
    BinaryTree *right;
};
```