Assignment 5: Can you guess my favorite character?

Description:

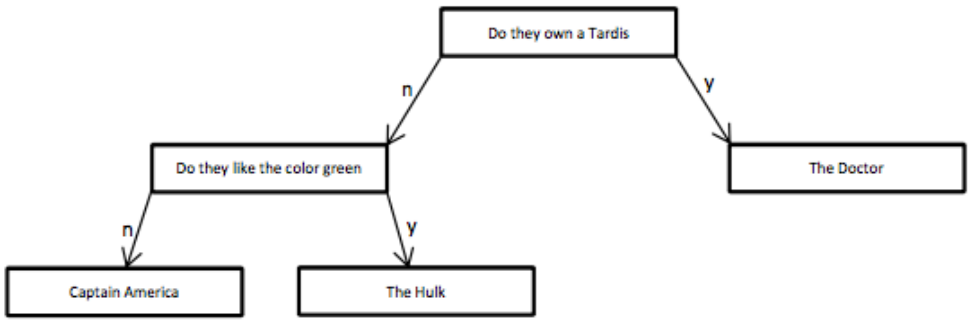
Use the knowledge we've gained about trees to build a program that is capable of guessing what a user is thinking. In this example, we will ask the user to think of their favorite cartoon/television/movie character and then attempt to guess what they are thinking by asking a series of questions. The program will use the answers to these questions to build out a decision tree and eventually be capable of guessing correctly.

Here is an example run:

```
$ ./a.out

Are you thinking of your favorite character? (y/n) y
Is it Captain America? (y/n) n
What is the character's name? The Doctor
What question would distinguish The Doctor from Captain America? Do they own a Tardis
If the character were The Doctor, what would the answer be? (y/n) y

Are you thinking of your favorite character? (y/n) y
Do they own a Tardis? (y/n) n
Is it Captain America? (y/n) n
What is the character's name? The Hulk
What question would distinguish The Hulk from Captain America? Do they like the color green
If the character were the Hulk, what would the answer be? (y/n) y

Are you thinking of your favorite character? (y/n) y
Do they own a Tardis? (y/n) n
Do they like the color green? (y/n) y
Is it The Hulk? (y/n) y
I rule!

Are you thinking of your favorite character? (y/n) n

$
```

As you can see, the program starts by guessing a random character and then learns from there. Eventually, it will be capable of guessing correctly after enough questions have been asked.

Simplistically, a decision tree is a binary tree where the left child represents no and the right child represents yes to whatever question may be contained within the parent node. For example, the decision tree for the previous example might look something like this:



The Program:

The program starts each round by asking the first question stored at the top (root) of the tree. Depending on the answer, it moves left or right until it reaches a leaf node. Once a leaf node is reached, it must make a guess. If the guess is not correct, it asks the user for the name of the new character and a question that distinguishes the wrong guess from the new character. It then adds a node to the tree with the new question and the new character.

Here is some pseudo code:

```
While the user is thinking of a character:

	Start at root

	While left child is not a leaf node:
		If user responds yes to question in current node:
			Follow right child
		Else:
			Follow left child

	Make a guess using current node

	If guess is correct:
		Tell the user you rule
		Move to next iteration of loop

	Prompt user for new character's name
	Prompt user for question that distinguishes new character

	Insert question into current node

	Prompt user what answer to question would be for current character

	If answer is yes:
		Set left child equal to guess
		Set right child equal to character
	Else:
		Set left child equal to character
		Set right child equal to guess
```

Requirements:

Write a class called CharacterTree that handles building and accessing your decision tree. The only requirement of this class is that it uses some sort of linear or dynamic array to store the tree itself. DO NOT USE STRUCTS. Refer to lecture on heaps for more information on representing a binary tree using an array.

Suggestions:

```
#include <string>
#include <vector>

getline( cin, … );

void CharacterTree::insert( string s, int index ){}
string CharacterTree::retrieve( int index ){}
```