CS/SE 1337 – Homework 5 – The Animal Guessing Game

Dr. Doug DeGroot's C++ Class

Due: April 2, 2023, by midnight

How to submit:

1. **Upload your .cpp file and a PDF of your program's output.** The program must compile on either Codeblocks or MS Visual Studio 2015 (or later). Don't upload an entire project folder – only your .cpp file and a PDF of the program's output.
2. Include your name in the file/project name, like so:
   HW5-CS1337-John-Doe

Maximum Number of Points: 100

Objective:

You are to write a program that plays the fun Animal Guessing game as we discussed in class. It will be much like the one I demonstrated in class but simpler. I hope you will find the game a fun one. Programs similar to this have been used as expert systems in many fields, so this is not just a "fun and games" project.

The main data structure we will use is a **binary tree of structs implemented with pointers**. In this binary tree we will store a database of animals and questions that can be used to guess an animal. The questions will be arbitrary yes/no questions, and names of animals will usually be one word but can be more.

When your program makes a guess, the guesses will be either correct or incorrect, again as demonstrated in class. Every time the program incorrectly guesses an animal type, the program must ask the user to help it learn the new animal type and ask how that new animal type can be differentiated from the other animal type that it guessed. The program will then update its database and be just a little bit smarter. The longer the user plays the game, the larger and more sophisticated the binary tree database will become, and thus the smarter the program will become.

The game ends when the program has learned about a new animal and stored the new animal type and the differentiating question or when it correctly guesses the user's animal. The program then asks the user to play the game again, and if the user agrees, the program starts all over but this time with a possibly augmented DB. (I know this is not the greatest explanation, so hopefully you paid attention during the discussions and demonstrations in class.)

Steps you are to code:

1. Define a structure such as the following:

```
struct animalNode {
    string question; //a question to ask the user
    string guess; //the name of an animal to guess
    animalNode* yesAnswer; //null or pointer to the “yes” animalNode
    animalNode* noAnswer; //null or pointer to the “no” animalNode
};
```

2. All animal struct nodes used by the program must be created dynamically (using the *new* command).
3. Each animal node will contain either a question to ask the user or the name of an animal to guess – *never both*! If the node contains a question, the guess member will be blank, and both the yesAnswer and the noAnswer will point to child animal nodes. If the animal node contains a guess, however, the question member will be blank, and both child pointers will be *nullptrs*.
4. Normally, you would want to read a file containing data that could be used to initialize the decision tree. (But that’s for later.) Right now, when the program starts, initialize the database by creating an empty *rootNode* and setting the guess member to “**lizard**.” Set both the yesAnswer and noAnswer to nullptrs and the question to the empty string. You will now have your first node, and you can begin playing the game.

Question: “”

Guess: lizard

yesAns: nullptr	noAns: nullptr

5. The game will first ask the user to think of an animal and wait for the user to hit the Enter key. Since the only node we have so far is a guess node, the program will then ask the user if the animal the user is thinking of is a lizard (that’s the value of the guess node). If the user enters “yes,” the program “wins,” and the game starts all over.
6. If the user enters “no,” then the program needs to ask the user to teach it about the new animal the user is thinking of. It will do this by asking the user 1) what animal was being thought of and 2) a question that can be used to differentiate that animal from the one the program guessed. (See the **example dialog** at the end of these instructions.)
7. At this point, you will need to grow the tree a bit and rearrange some of the information in it. You will currently be pointing at a guess node in the database; let’s call this one *curNode*. It will by necessity contain a guess and not a question, right? Start by creating two new animalNodes; point to one of them with the yesAns member and to the other with the noAns member.

8. Then ask the user to enter a question that can be used to differentiate the current animal from the new animal. Read that question in and store it in the current node's question member; then clear the current node's guess member.
9. Ask the user whether the answer to the question for the new animal is "yes" or "no,", and depending on the answer, store the new animal's name in either the yesAns node or the noAns node.
10. Finally, move the previously guessed animal's name (the one that was in curNode) to the other animalNode's guess member and clear both children pointers.
11. Then start the game all over, using the newly modified binary tree database.

12. **NOTE:** You don't actually need two different strings for each animal node to store the question or a guess. You could instead use just one string, maybe called QorA or something similar, to store either the question to be asked or the animal to be guessed. Somehow, I thought that might confuse you more than using two separate strings. But if it doesn't, I'd suggest using a single string. How can you tell whether the string QorA holds a question or a guess? Simply by checking to see whether *either* the yesAns or the noAns is non-null. Non-null values would dictate that the node contains a question; nulls would dictate that the node is a guess.

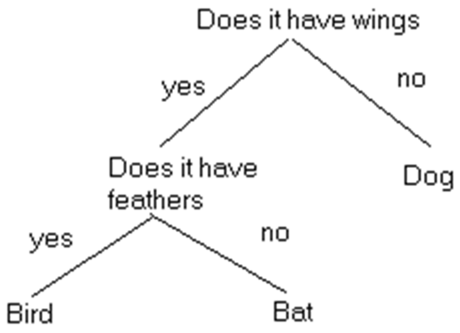
13. **NOTE:** Remember that you can make this same program applicable to any number of different domains, e.g., movies, songs, illnesses, books, or whatever. You might want to consider adding capabilities to your program to make it more generic so that it can handle a number of different domains. (Not just a different one, but a number of different ones.)
14. **You must add at least 10 different animals to your execution.** Include a printout with your homework.
15. You might want to review my files on creating binary trees, inserting a node into a linked list, deleting a node from a linked list, etc. that are up on eLearning.

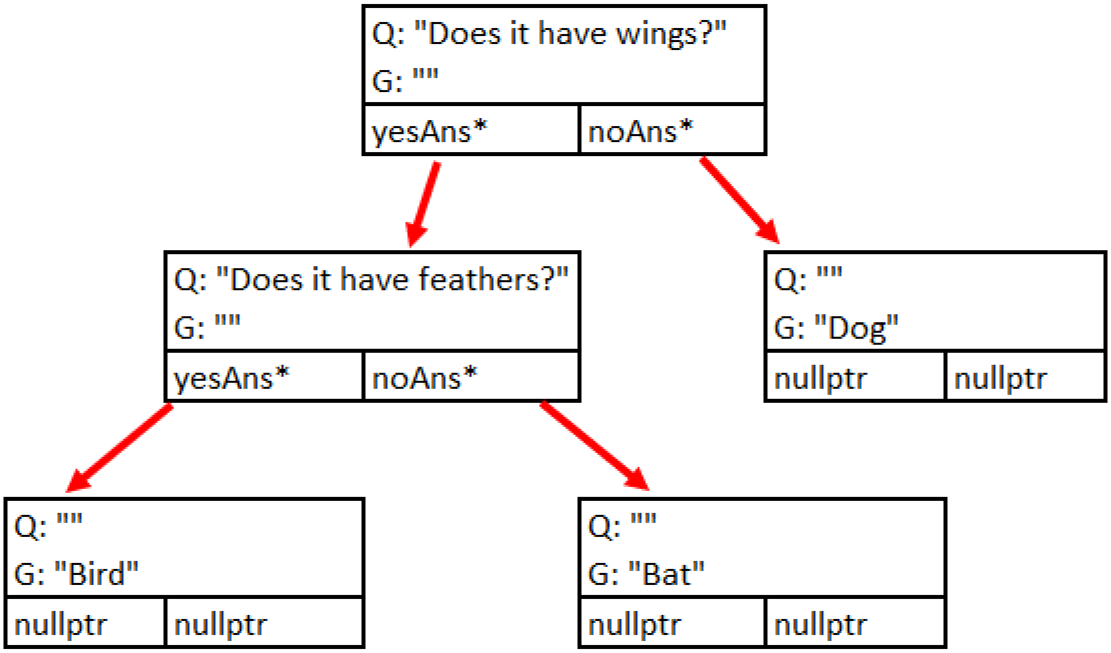
Annotations for the code:

1. The main function can be at either the beginning or the end of the program. I don't care which.
2. Add comments at the top of your main.cpp file to include your name, the name of the program, and notes on how your design works when executed. Point out any special features or techniques you added using a comment saying **"// Special Features**."
3. Comment your code effectively, as we continue to discuss in class. Use descriptive variable names everywhere so that the code becomes as self-documenting as possible. Use additional commentary to improve readability and comprehensibility by other people. But don't add superfluous comments; use good, self-documenting code instead.
4. You absolutely MUST use consistent indentation and coding styles throughout the program. Failure to do so will result in a loss of five points.
5. If the program does not work at all, or works incorrectly, 10 points will be deducted.

6. 10 points will be deducted from your final grade for each day the homework submission is late. No exceptions. After three days late, a zero will be assigned as the grade.

An Illustration:





Example Dialog:

```
Let's play the "Guess the Animal" game.
Think of an animal and hit return when you’re ready.

Does it have cubs for babies?: y
Does it perform in the circus?: y
Is it a(n) Tiger?
y
Good! I guessed your animal.
Try again?
: y

Does it have cubs for babies?: y
Does it perform in the circus?: n
Is it a(n) bear?
y
Good! I guessed your animal.
Try again?
: y

Does it have cubs for babies?: n
Does it swim?: y
Did it swallow Jonah?: y
Is it a(n) whale?
y
Good! I guessed your animal.
Try again?
: y

Does it have cubs for babies?: n
Does it swim?: n
Does it eat cheese?: n
Does it purr?: n
Is it afraid of mice?: n
Is it often nicknamed Billy?: n
Is your animal a(n) lizard? ← we are a leaf of the tree, a guess node, so make the guess
n
Bummer! What animal were you thinking of?
giraffe
What is a yes/no question that I can use to tell a lizard from a giraffe?
Does it have a long neck?
For a giraffe, is the answer yes or no?
yes
```

```
Try again?
: n
```

The program would quit at this point

Example Starting-Out Dialog:

```
Let's play the "Guess the Animal" game.
Think of an animal and hit return when you’re ready.

Is it a(an) lizard?
No
Bummer! What animal were you thinking of?
a dog
What is a yes/no question that I can use to tell a lizard from a dog?
Does it have a long, slithering tongue?
For a dog, is the answer yes or no?
no
Try again?
: n
```