# CS 211 PA #4: Guessing Game

In this assignment, you will build a "yes/no" animal guessing game using a binary tree. In this scenario, each tree node represents a question with the left child representing the response to a "no" and the right child representing the response to a "yes." In building this structure, you will find that you can quickly build an "intelligent" guesser. Here's the basic flow of the game:

### Until we reach a leaf node:

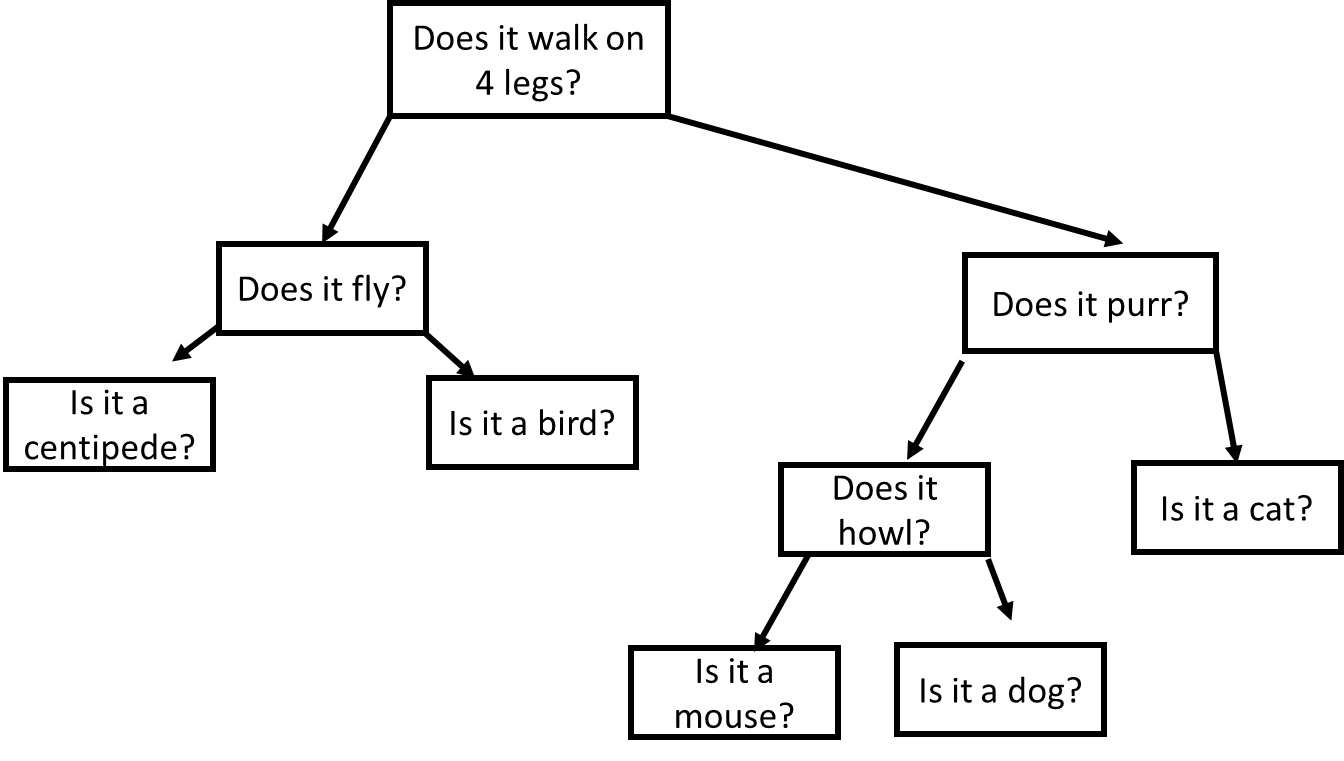
1. Output the value of the node (i.e. next question to ask, "e.g. does it have fur?")
   1. If the user's response is "yes", pull the next question from the right child
   2. If the user's response is "no", pull the next question from the left child

### Once at a leaf node:

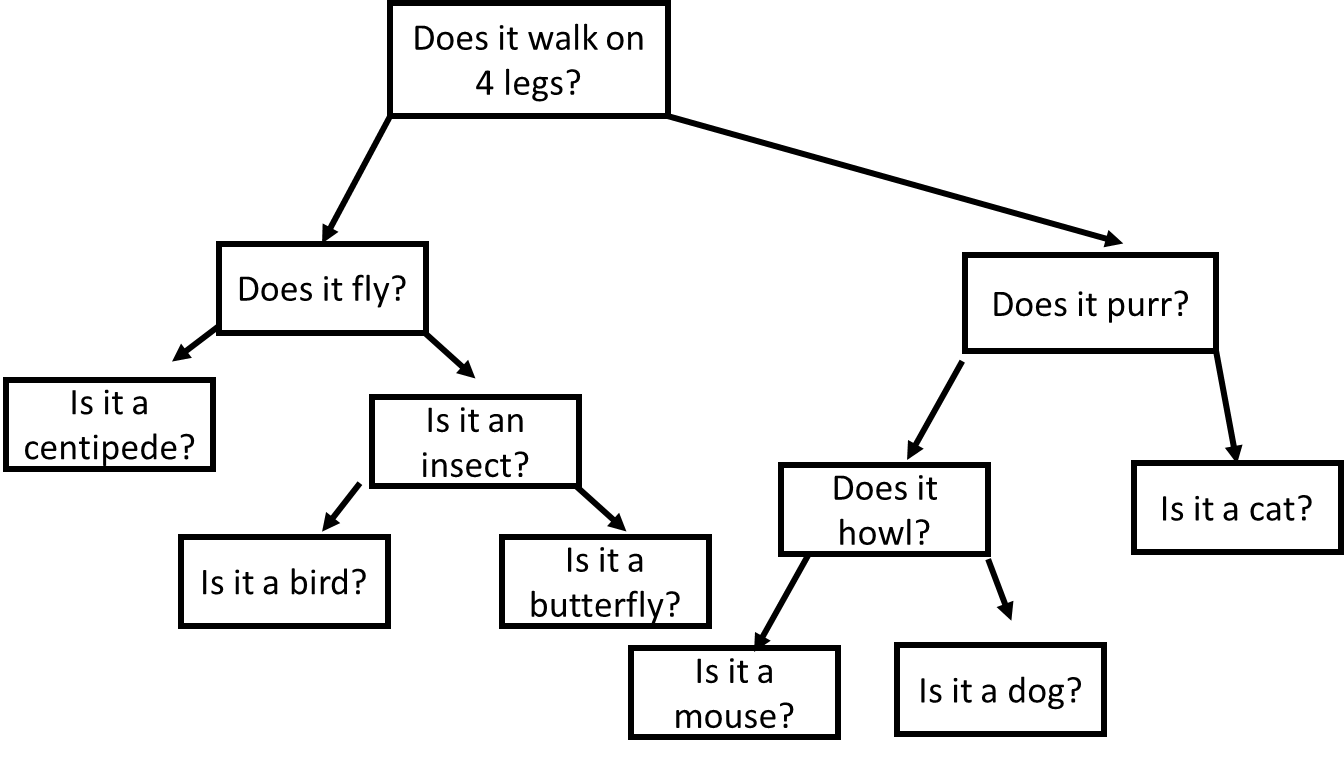
1. Read the value of the leaf node, which should be a question (e.g. "is it a dog?")
   1. If the answer is yes, output that you win and that the game is over
   2. If the answer is no:
      1. Ask the user for the correct answer (e.g. "cat")
      2. Ask the user for a new question, "What is a question that I can use to differentiate between my guess and your answer" (example response, "does it purr?")
      3. Remember the question and the answer, restructure tree accordingly

## Graphical Example

The textual example might be a bit hard to follow, so consider the following tree:



In the example above, the game always starts by asking whether or not the animal walks on 4 legs. If the answer is yes, it will next ask if the animal purrs. Otherwise, it will ask if the animal flies. If the answer is yes, the game will guess bird because that is a leaf node. If that is the correct answer the game is over. However, if we instead were thinking about a butterfly, the game would prompt us for the correct answer and a differentiating question. Thus, after one game, our tree might look like the following:



As you will soon see, your game starts to become very smart very quickly. How quickly can you get it to guess most animals given to you by friends?

## Storing your tree

In order to build large trees, you will need to save and load your tree from a file. The easiest way to do this is to perform a pre-order traversal and write the results to a file. For example, a pre-order walk of the tree above would produce the following file:

|  |
| --- |
| Does it walk on 4 legs?  Does it fly?  \*centipede?  Is it an insect?  \*bird?  \*butterfly?  Does it purr?  Does it howl?  \*mouse?  \*dog?  \*cat? |

Notice how I mark leaf nodes with asterisks (\*). If we do not do so, we would be unable to reconstruct the tree as we wouldn't know when to stop recursively building the tree. Also note that I only store the animal name as the "Is it a" part is generated by my program and doesn’t need to exist outside of that context.

## Example Run

Below, I give you example runs of the program:

### Run #1

|  |
| --- |
| \*\*\*Animal Guessing Game\*\*\*  Load file? (yes/nno): no  Save results to: game1.txt  Is it a dog? (yes/no): No  Darn, I got it wrong. What is the correct answer: bird  What is a yes/no question that I can use to differentiate between a dog and a bird?:  Does it fly?  Is the answer to this question yes or no?: yes  Would you like to play again? (yes/no): no  Game over |

### Run #2

|  |
| --- |
| \*\*\*Animal Guessing Game\*\*\*  Load file? (y/n): y  File Name: game1.txt  ...game1.txt loaded  Does it fly? (yes/no): no  Is it a dog? (yes/no): no  Darn, I got it wrong. What is the correct answer: cat  What is a yes/no question that I can use to differentiate between a dog and a cat?:  Does it purr?  Is the answer to this question yes or no?: yes  Would you like to play again? (yes/no): no  Game over |

## Starter Code

There is no starter code provided for this project – you are free to implement the program however you see best. That being said, it might be useful for you to reuse my BinaryNode class as it provides a solid foundation for any binary tree.

## Possible Strategy for Getting Started

1. Note that you'll need a default animal whenever you start a fresh game. I suggest something common, e.g. "dog."
2. Polymorphism can be used to distinguish between questions and answers. For example, you could derive a QuestionNode and an AnswerNode from a base BinaryNode.
3. Don't worry about saving to the file. Get the tree constructed and the guessing game working before you think about saving and loading your tree.

## Header Comment, and Formatting

1. Be sure to modify the file header comment at the top of your script to indicate your name, student ID, completion time, and the names of any individuals that you collaborated with on the assignment.
2. Remember to follow the basic coding style guide. A basic list of rules is included with this document.

# Reflection Essay

In addition to the programming tasks listed above, your submission must include an essay that reflects on your experiences with this homework. This essay must be at least 350 words long. Note that the focus of this paper should be on your reflection, ***not*** on structure (e.g. introductory paragraph, conclusion, etc.). The essay is graded on content (i.e. it shows deep though) rather than syntax (e.g. spelling) and structure. Below are some prompts that can be used to get you thinking. Feel free to use these or to make up your own.

* Describe a particular struggle that you overcame when working on this programming assignment.
* Conversely, describe an issue with your assignment that you were unable to resolve.
* Provide advice to a future student on how he or she might succeed on this assignment.
* Describe the most fun aspect of the assignment.
* Describe the most challenging aspect of the assignment.
* Describe the most difficult aspect of the assignment to understand.
* Provide any suggestions for improving the assignment in the future.

## Deliverables

You must upload your program and reflection as a ZIP file through Canvas no later than midnight on Friday, March 30, 2018.

## PA #3 Checkins

During lab on 3/26, you must demonstrate a program that is able to ask at least two questions and make at least one guess.

## Grading Criteria

Your assignment will be judged by the following criteria:

### Reflection essay (5pts)

* Your reflection meets the minimum requirements as specified earlier in this document.

### PA Checkin (10pts)

* You successfully demo your code during lab.

### Style (10pts)

* Your project contains good structure and implements the required classes. Your program intelligently uses classes when appropriate and generally conforms to good OOP design (i.e. everything isn't slapped into main).

### Dynamic Memory Management (5pts)

* You dynamically allocate memory in your program.
* You correctly clean up all dynamically allocated memory in your program.

### Guessing Game (30 pts)

* Your program correctly plays the guessing game.
* Your program is able to incorporate new responses into the game.

### Saving and Loading of Games (20 pts)

* Your program correctly saves a tree to file (10pts)
* Your program correctly loads a tree from a file (10pts)