PIC 10B Lectures 1 and 2 Spring 2015
Homework Assignment #7

Due Wednesday, May 27, 2015 by 6:00pm.

Objectives:

1. To gain experience using trees to organize data and simulate artificial intelligence.

Introduction:

This problem is adapted from *Data Abstraction and Problem Solving in C++*, by Carrano, Helman, and Veroff.

A **binary tree** is a tree whose nodes have at most two children. In this assignment, you will be defining a class called Learner which will use a BinaryTree object (not a BinarySearchTree) member to help it remember how to recognize various animals.

The learner initially knows how to distinguish between a dog and a snake by the question "Does it have legs?" The binary tree will initially have that question as its root node. The right answer "dog" is the question's left child and a wrong answer "snake" is its right child.

The Learner tries to guess the animal the user is thinking of by asking a sequence of questions, starting at the root question "Does it have legs?". Depending on the user's answer, it will try to ask another question to help it distinguish among all the animals it knows are correct answers to that question.

For any given question in the binary tree, all of the animals the Learner knows which are "yes" answers are to the left of the question and all of the animals the Learner knows which are "no" answers are to the right. In the tree, questions are all internal nodes and the animals are all leaves (have no children).

The BinaryTree class will have the following `private` members:

- `root` - a TreeNode*
- `bool isLeaf(TreeNode* node)`
  - A helper function that determines whether the given TreeNode node is a leaf. It returns true if it is a leaf and false otherwise. It does not change the calling object.
- `void free(TreeNode*& subtree)`
  - Uses postorder traversal to print the data values of and all deallocate the TreeNode objects (one at a time) in the given subtree.

- `void display(ostream& os, TreeNode* subtree)`
    - Send the values of the TreeNodes in the given subtree one at a time to the given output stream using a preorder traversal. It does not change the calling object.

The BinaryTree class will have the following `public` members:

- `BinaryTree()`
    - Default constructor that creates an empty BinaryTree object.
- `~BinaryTree()`
    - Destructor that uses postorder traversal to print the data values of and all deallocate the TreeNode objects (one at a time) in the tree.
- `void display(ostream& os)`
    - Send the values of the TreeNodes in the tree one at a time to the given output stream using a preorder traversal starting at the root. It does not change the calling object.

The BinaryTree will also overload the output `operator<<` as a `friend`. Use the appropriate member function of BinaryTree above to implement it.

The class Learner will have the following `private` members:

- `memory` - a BinaryTree object holding the Learner's knowledge.
- `void learn(string question, string yes_answer, string no_answer, TreeNode* subtree)`
    - a helper method that overwrites the given data member of subtree (assuming subtree is not NULL) with the given question and makes the given yes_answer a left child and the given no_answer a right child.

The class Learner will be a `friend` class of BinaryTree and will have the following `public` members:

- `Learner()`
    - Default constructor that initializes its BinaryTree memory to have one initial question with yes and no answers.
- `void learn()`
    - This member does all the work for the Learner class. It displays the "Let me guess.." message and asks the user a series of questions in the memory BinaryTree until it hits an animal TreeNode. It then asks the user if the answer was that animal. If yes, tell the user "I win". Otherwise the Learner must learn a new question and a new animal that the user provides. Allow the user to keep thinking of animals for the Learner to guess until they wish to quit. It should then tell the user what the Learner learned by displaying its memory tree.

Directions:

1. Create the project called “Hw7” in a solution called “Homework” using Microsoft Visual Studio 2012. All header (.h) and source (.cpp) files inside the project should contain the following header:

```
/*
   <Your Name>                    PIC 10B Intermediate Programming
   ID: <Your Student ID>          Spring 2015
   Email: <Your Email Address>    Assignment #7
   Section: <Your Section # eg 1A>
   Honesty Pledge:

      I, <Your Name Here>, pledge that this is my own independent work,
   which conforms to the guidelines of academic honesty as described in
   the course syllabus.

   List of known bugs:  <Known bugs, if any>
*/
```

2. Include the given files TreeNode.h and and TreeNode.cpp files posted on CCLE’s Homework #7 Files in your Hw7 project.

3. Define the BinaryTree according to the specifications above and include its header and implementation files in your project.

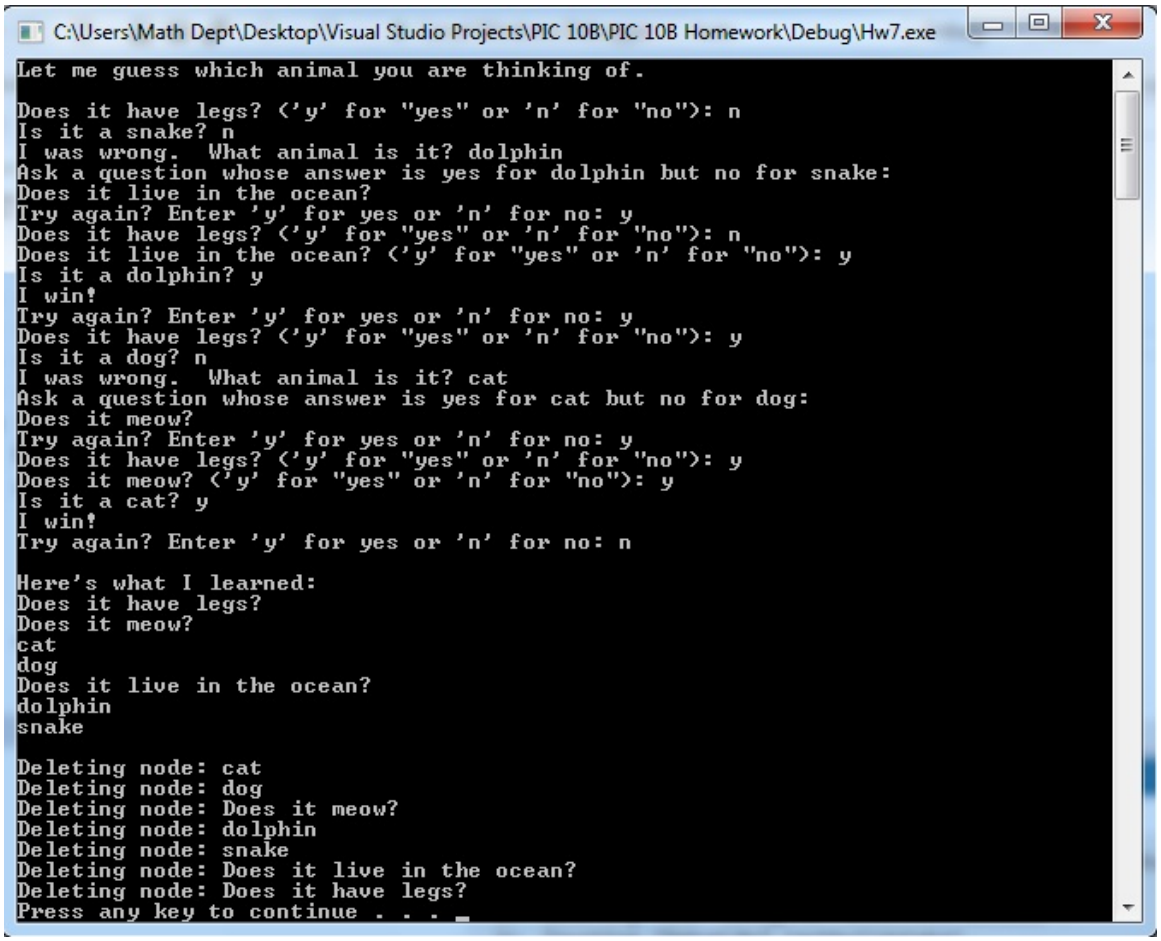
4. Define the Learner class according to the specifications above and include its header and implementation files in your project.

5. Include the given application file SchoolApp.cpp posted on CCLE’s Homework #7 Files in your Hw7 project.

6. When you have completed your project, be sure to
   - make sure your program compiles in Visual Studio 2012.
   - run your program to make sure it works correctly
   - upload your source code files
     - `BinaryTree.h`
     - `BinaryTree.cpp`
     - `Learner.h`
     - `Learner.cpp`

     using the CCLE website. No hardcopies will be collected.
   - visually verify that your source code was submitted correctly by clicking on the links to those files on the CCLE page after submission.

Here is a sample screenshot of your application:



Grade Breakdown:

Criteria	Description	Points
Header	Starts every .h and .cpp file	1
Comments	Program well-commented.	1
BinaryTree.h	Class defined correctly.	2
BinaryTree.cpp	`BinaryTree(), ~BinaryTree()`	2
	`free`	3
	`display` functions	3
	`isLeaf`	2
	`operator <<`	2
Learner.h	Class defined correctly.	2
Learner.cpp	`Learner()`	2
	`learn(string, string, string, TreeNode*&)`	2
	`learn()`	8
Total		30

A penalty of 5 points will be assessed if your code does not compile using Microsoft Visual Studio 2012.