# GDAPS2 – Practice Exercise

# Talent Tree Traversal

## Objective:

Gain familiarity with building trees manually, and recursively traversing them. This will be a *console application*.

## Details:

A *talent tree* is a way of organizing the prerequisite abilities that are necessary to learn a new ability in a game. If each ability has at most one direct prerequisite ability, and all abilities are connected, then it is also a tree in the computer science sense: a set of connected *nodes* with no *cycles* (loops). If one ability is a prerequisite for all the others, it is the *root* of the tree; if a single node has multiple abilities that it enables, it is the *parent* of those *children.*

## Example:

Magic

Magic Arrow

Fireball

Exploding Arrow

Ice Arrow

1000 Tiny Fireballs

Crazy Big Fireball

## Coding:

Design a TalentTreeNode class for a “talent tree” data structure. It should have the following fields:

* The ability name
* Boolean indicating if the player has learned this talent
* Exactly two TalentTreeNodes: a left child node and a right child node

Make an appropriate constructor for a TalentTreeNode object, which can set both its name and whether the player has learned it. You will also need a way to get & set each child node.

In your main method, create a sample talent tree of at least depth 3 (that is, three levels) with some nodes learned and some unlearned (make sure it’s legal – no skipping talent prerequisites). You’ll have to do this manually: Create all seven nodes in temporary variables, and hardcode their relationships. It may help to draw out the tree.

Write the following recursive methods in TalentTreeNode, then call them on only your root node to show that they work:

* *ListAllAbilities( )* should print all of the abilities in the tree using “in order” traversal
  + *Recursion*: Call the method on the left node (if it exists)
  + Print this one
  + *Recursion*: Call the method on the right node (if it exists)
* *ListKnownAbilities( )* should print out which abilities the player knows.
  + *Recursion:* Print “this” node’s ability if it has been learned, then check for any child nodes and call their ListKnownAbilities().
  + If the current node’s ability has not been learned, there’s no need to check for child nodes, since they couldn’t have been learned yet anyway.
* *ListPossibleAbilities( )* should only print the abilities the player could learn *next*. These are all the abilities that are not yet known, but have a parent that is known.
  + *Recursion:* Only print a node’s ability if it is not known, but its parent’s ability is known.
  + To do this, you’ll need to figure out the condition for printing and the condition for recursion. If you get both correct, it should only print abilities that have not been learned, but whose parents HAVE been learned.
  + Note: A node does NOT need to know about its parent for this to work.

## Finished?

**Make sure you follow the coding standards for all code you create.**

Demonstrate your code to the instructor or TA and show them the code.