**CS223 - Lesson 18 – worksheet on tree implementation:**

In the tree above, a node is either an operator (+, -, \* / etc.) or an operand (a numerical value). If a node is an operator, its value must be determined before its value can be used by its parent operation. The tree, or any of its sub-trees, always evaluate to a single numerical value.

1. Write the equation this tree represents. Make sure you use parentheses to guarantee the correct order of operations. (Note that the tree does not need to include the parentheses. Why not?)
2. Create a binary tree that would represent this equation:

(2 – 12) + 8 \* (10 - 5)

Using C typedef statements, define the declarations you would need to represent a tree that stores mathematical equations (as in the previous page). Make sure you include a definition for "Tree".

1. Your definition uses a "pointer implementation," where each node stores the same data type (string).
2. Your definition uses a "pointer implementation," where each node stores possibly a different data type (either a character that represents an operator (+, -, \* / etc.) or an integer that represents an operand.
3. Your definition uses an "array implementation," where each node stores the same data type (string).  
   * 1. What position of the array holds the root node?
     2. How big does the array need to be to hold an equation that is represented with a tree that has a height of 3?
     3. Write an appropriate malloc statement to allocate the array for a tree that has a height of 3.

* + 1. Given the array index, 3, where is this "node's" left child node?
    2. Given the array index, 3, where is this "node's" right child node?
    3. Given the array index, 3, where is this "node's" parent node?

1. Your definition uses an "array implementation," where each node stores possibly a different data type.
   * 1. Write an appropriate malloc statement to allocate the array for a tree that has a height of 3.
2. You have defined 4 different implementations for a tree that stores equations. Assume that your "pointer implementations" only store nodes that contain data, while your "array implementations" must store an array large enough to store a full and complete binary tree. Calculate the amount of memory each implementation uses. Assume your equation requires a tree of height 3 and has 8 nodes.  
   1. Pointer implementation (same data type for every node)
   2. Pointer implementation (different data type for every node)
   3. Array implementation (same data type for every node)
   4. Array implementation (different data type for every node)
3. Implement your definitions in a C program and verify that your typedef's and calculations are accurate. (Print out sizeof() and memory addresses to determine memory usage.)
4. You want to develop a ternary tree using an array, where each node can have 0, 1, 2, or 3 children. Discover the formula's that you would use to calculate the indexes of child and parent nodes. (Hint: Draw a complete and full ternary tree. Label the nodes starting at 1 (root) and going left to right over each level. Then study the pattern and try various formulas until one works for all cases.)

1. Discover the formula's that you would use to calculate the indexes of child and parent nodes for a quad tree. (Each node has up to 4 children.)