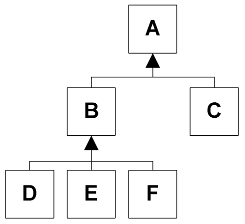
# Tree Representation and Parsing Test

## Introduction

This test provides you the opportunity to demonstrate your knowledge of data structures and algorithms.

## Problem



A tree is shown to the right. Arrows point from sub-trees to their parents. Boxes represent nodes in the tree. Each node has a unique, single-character identifier.

Question 1: How might you represent this tree as a sequence of characters?

Question 2: How much storage (in characters) does your representation take?

Question 3: Give a formula for the number of characters your representation would require for complete (full) binary trees (branching factor b = 2)? Hint: the formula should be a function of N, the number of nodes in the tree.

$F_2(N) =$

Question 4: Give a formula for the number of characters your representation would require for complete (full) trees an arbitrary branching factor b. Hint: the formula should be a function of N, the number of nodes in the tree and b, the branching factor.

$F(N,b) =$

Question 5: Is your representation "good"? Are there better representations?

Question 6: Construct an algorithm in a language of your choice to parse your tree into an in-memory structure.

```
    parse( r:Reader):Tree
```

Question 7: Construct an algorithm that will traverse the in-memory tree and render as a stream of characters.

```
    enumerate( t:Tree, s:Writer):void
```

Extra Credit: Starting with the algorithm from question 7 (tree walker), can you alter it so that it lazily enumerates the nodes of the tree.

```
    lazily_enumerate( t: Tree, w:Writer, e:Enumeration): Node
```

Extra Credit: Write an algorithm to compare two trees for equality. Two trees are equal if they have the exact same shape and they have nodes with the exact same names at the same relative positions in the trees.

```
    compare( lhs:Tree, rhs:Tree):Boolean
```