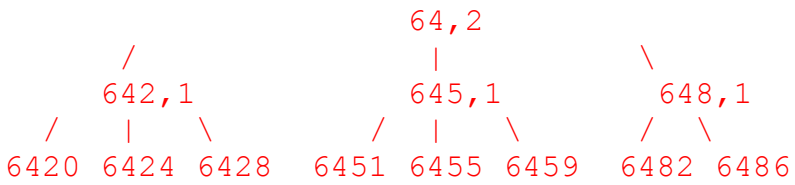
3) (10 pts) ALG (Backtracking)

We define a number as Digit Sum Divisible if, for each value of i, the sum of its first i digits is divisible by i. For example, the number 6451 is Digit Sum Divisible because 6 is divisible by 1, 6 + 4 = 10 is divisible by 2, 6 + 4 + 5 = 15 is divisible by 3 and 6 + 4 + 5 + 1 = 16 is divisible by 4. Consider writing a backtracking function that outputs all Digit Sum Divisible numbers of a particular length given a particular prefix. This function would take in a prefix, such as "64" and a number of digits left to add to it (for this example, 2), and the function would print out each Digit Sum Divisible number starting with the digits 64 that are four digits long. The function would use backtracking because instead of adding each possible digit to the given prefix and making a recursive call, it would first check to see if doing so would maintain the divisibility requirement for the next length. (In this example, 640 would be skipped since 6 + 4 + 0 = 10 and 10 isn't divisible by 3.) ***Write out a tree structure that shows each unique prefix that occurs for each recursive call for the specific function call with the prefix 64 and 2 digits left to add to it.*** (Note: The root node of your tree should be 64, each child of 64 should be a three digit number, and each child of those children should have a four digit number. There should be eight leaf nodes in the tree, so make sure to leave enough room for eight nodes at the bottom level. These leaf nodes are the eight numbers the function would print out for this specific call.) **Note: Please do NOT write any code for this problem, just write out the underlined specified task above.**



(all calls on the last level have a k=0 with them)

Grading: 1 pt for level 0, 1 pt for level 1, 1 pt for each item in level 2. If extra items are included, subtract 1, cap at 0. Note: No need to add the value of k for each recursive call in the tree. They are printed in this solution for clarity's sake.