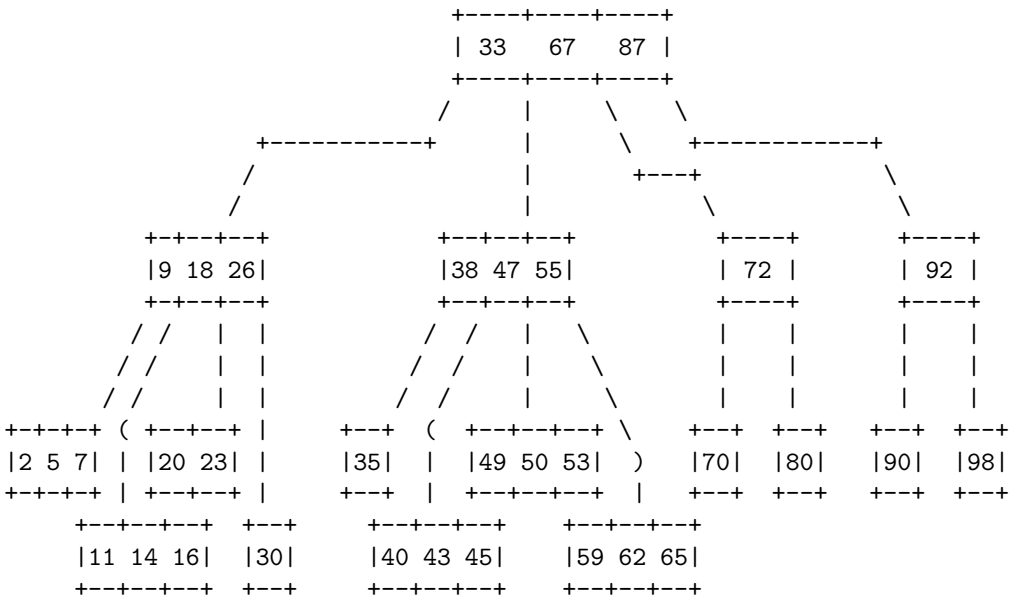
CMPS340 File Processing
HW #5 Fall 2016
Due: 1:00pm, Tuesday, October 25

Consider the B-tree T of order 4 illustrated below. (Recall that "order 4" means, in part, that each node must contain at least one key but no more than three.) For each operation in the list **(a)** through **(h)**, show the B-tree that results from performing that operation on T. (Each operation is to be applied to the tree T illustrated below, *not* to the tree resulting from applying all the previous operations to T.) You need not draw the entire tree each time—just show that portion of it that was changed in carrying out the operation, as well as a little surrounding context.

Assume, in carrying out the operations, that *redistribution* is used whenever possible. That is, split an overflowing node only if all of its adjacent/immediate siblings are full. Similarly, concatenate/merge two nodes only if all adjacent siblings of the underflowing node are on the verge of underflowing.[1] Follow the algorithms presented in class, which correspond to those described on the relevant web page. Keep in mind that redistribution involves *two* adjacent siblings and their parent, *not* three or more siblings and *not* first (or second, etc.) cousins (i.e., nodes with a common grandparent, great-grandparent, etc.).

(a) insert 22	**(c)** insert 42	**(e)** insert 51	**(g)** insert 4
(b) delete 20	**(d)** delete 67	**(f)** delete 80	**(h)** delete 90



[1]Not every node has two adjacent siblings, of course. A node that is the leftmost or rightmost child of its parent has only one, and the root has none.