2. The statement is true. For Approach 1, if a node has children, then it has pointers pointing out, and all nodes except the root node have pointers that point back to their parent. For Approach 2, if a node has children, it has only 1 pointer point out to one of its children. Still, all nodes except the root node have pointers point back to their parent. So for these 2 Approaches, the number of pointers pointing back is the same.

For Approach 1, when a node has n children, it has n pointers pointing out, and each child has a pointer that points back to it. Which means for any node, for each pointer that's pointing out, there must be a pointer that points back to it. So, for a tree with n + 1 nodes, it has n nodes besides the root. And there are n + n = 2n pointers in total.

For Approach 2, no matter how many children a node has, it has at most 2 pointers. 1 pointer points out to the left-most child, and 1 pointer points out to its right sibling, if it has one. But the number of points that points back to it is the same with Approach 1, which is determined by the number of children it has, which is n pointers. Amongst the n child nodes, there exist n – 1 pointers. So, for a node with n children, there is 1 pointer going out and n – 1 pointers amongst the child nodes. And this is the same for every height of the tree. So, for a tree with n + 1 nodes, it has n nodes besides the root. And there are 1 + (n – 1) + n = 1 + n – 1 + n = 2n pointers. And 2n = 2n for Approach 1 and Approach 2. Hence, the total number of pointers we need with Approach 1 is the same as the total number of pointers we need with Approach 2.