CSC 212: Data Structures and Abstractions
Spring 2018
University of Rhode Island

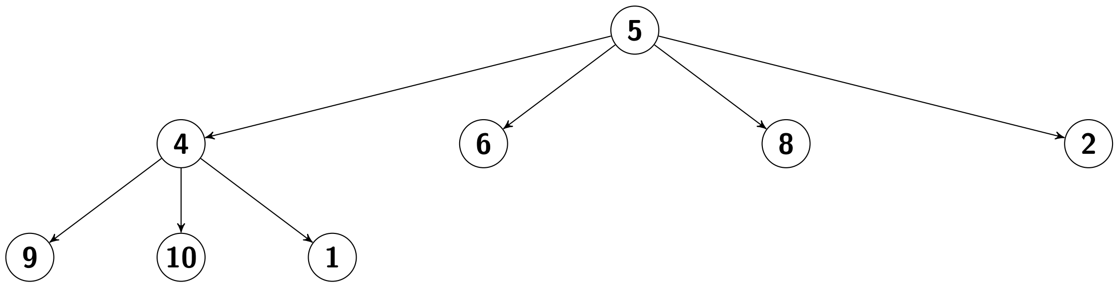
Weekly Problem Set #10 Solutions

Due Thursday 4/12 before class. Please turn in neat, and organized, answers hand-written on standard-sized paper **without any fringe**. At the top of each sheet you hand in, please write your name, and ID. The only library you're allowed to use in your answers is `iostream`.

1 K-ary Trees

1. Draw a k-ary tree, where `k=4`, after the insertion of the following elements in order: *Assuming insertions are performed left to right, level by level*

   `[5, 4, 6, 8, 2, 9, 10, 1]`



2. Looking at the tree you have drawn, how many leaves and nodes are present?

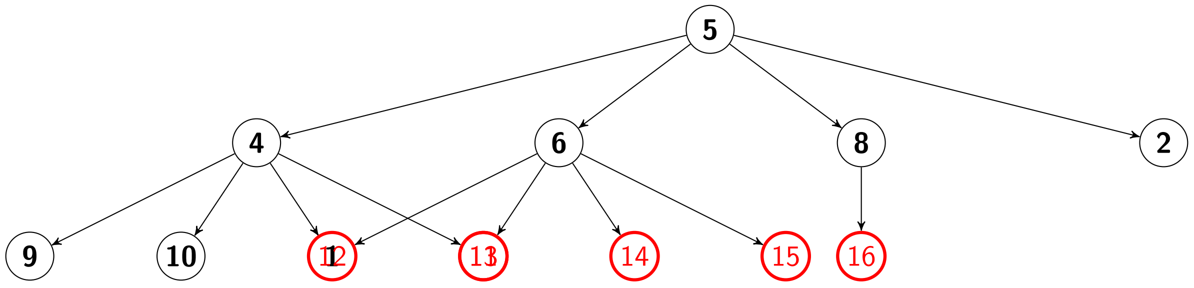
   6 leaves, 8 nodes, 2 of which are internal.

3. Examine your tree and find both the root and the node with the value 4. For both nodes, list the following attributes: depth, height of subtrees, number of siblings, number of children.

   Root Node: { depth: 0, height of subtrees: [2, 1, 1, 1], number of siblings: 0, number of direct children: 4, descendants: 7 }

   Node(4) { depth: 1, height of subtrees: [1, 1, 1, 0], number of siblings: 3, number of direct children: 3, descendants: 3 }

4. Insert 6 more random elements into your tree and relist any of the above attributes that have changed.



   Root Node: { depth: 0, height of subtrees: [2, 2, 2, 1], number of siblings: 0, number of direct children: 4, descendants: 13 }

   Node(4) { depth: 1, height of subtrees: [1, 1, 1, 1], number of siblings: 3, number of direct children: 4, descendants: 4 }

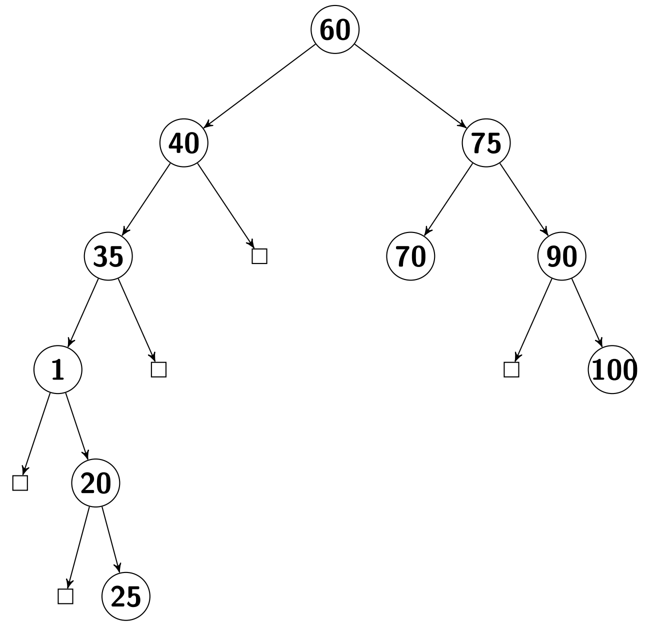
5. Would the structure of the k-ary tree you've drawn change at all if the elements were inserted in sorted order? Explain why or why not.

   No. K-ary trees are not sorted trees.

2 Binary Trees

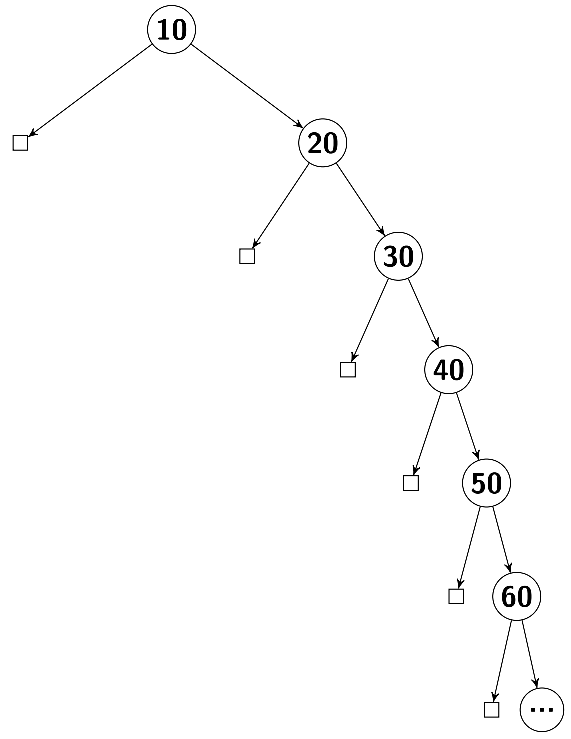
1. Draw a binary tree after the insertion of the following elements in order:

   `[60, 40, 35, 75, 90, 1, 20, 100, 25, 70]`



2. Draw a binary tree after the insertion of the following elements in order:

   `[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]`

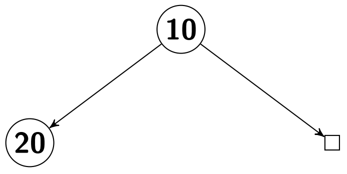


3. Explain what differs in the above two trees. Specifically, address how many leaf nodes there are, the depth of the right, and leftmost nodes, and the height of both the left and right subtrees from the root node.

   The tree with elements inserted in sorted order only has one leaf, whereas the previous tree had 3. The sorted tree is also extremely imbalanced, with all elements going towards the right.

4. If a Binary Tree is complete, does that necessarily mean it is also full? Justify your answer with drawings of trees.

   No, the deepest layer of the tree could have a node that only possesses a single child.

   Is complete, but not full.

3 Stacks and Queues

1. Continuing from your previous `LinkedList` solution, what changes would be required to convert this list into a generic queue? What about a stack?

   It would require modifying the API, simplifying the interface to allow fewer, operations.

2. Is a linked list the best underlying structure to implement a queue with? Justify your answer.

   Linked lists are a convenient underlying structure, but there is no true best when it comes to queues. So long as your implementation offers constant time operations and minimal space requirements.

3. Implement both a queue, and a stack. Provide only the essential methods for both (Constructor, Destructor, Push, Pop, Peek), and use whatever underlying structure you would prefer.