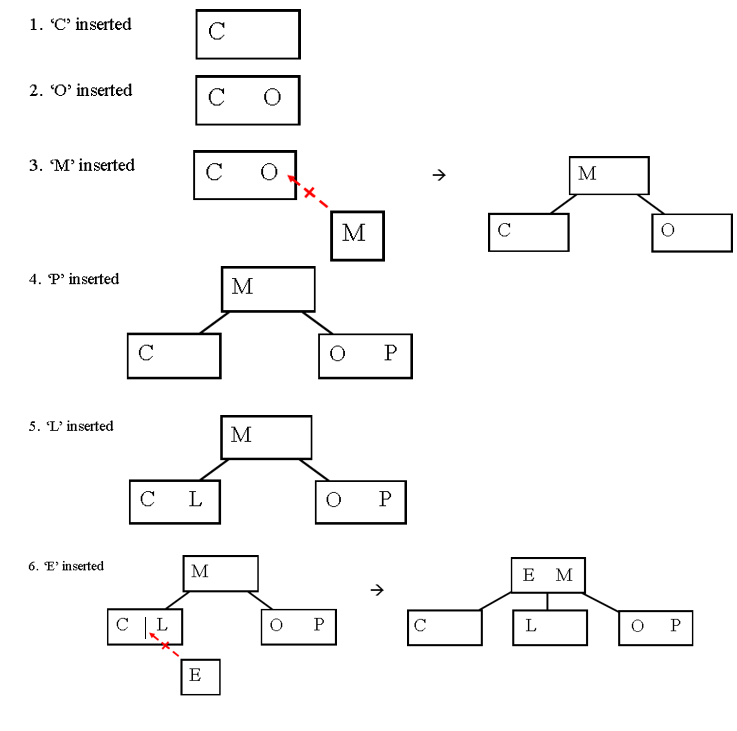
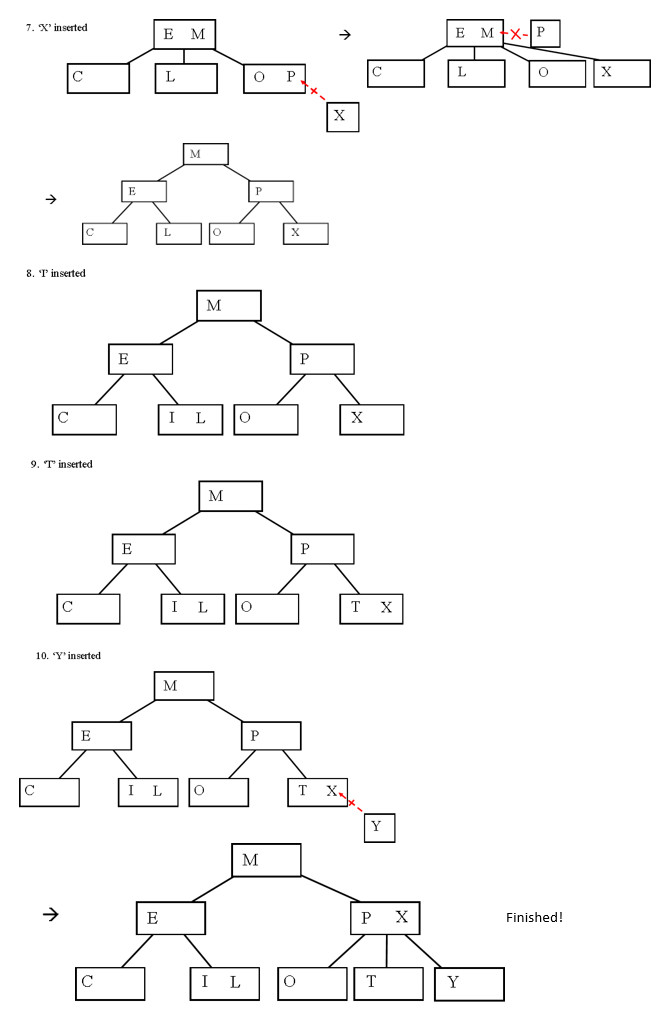
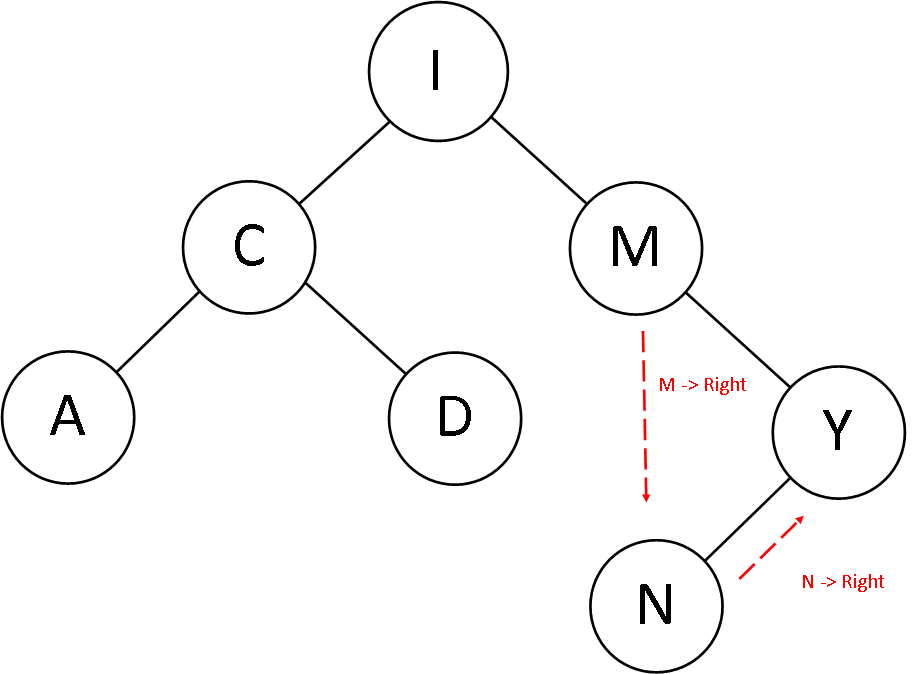
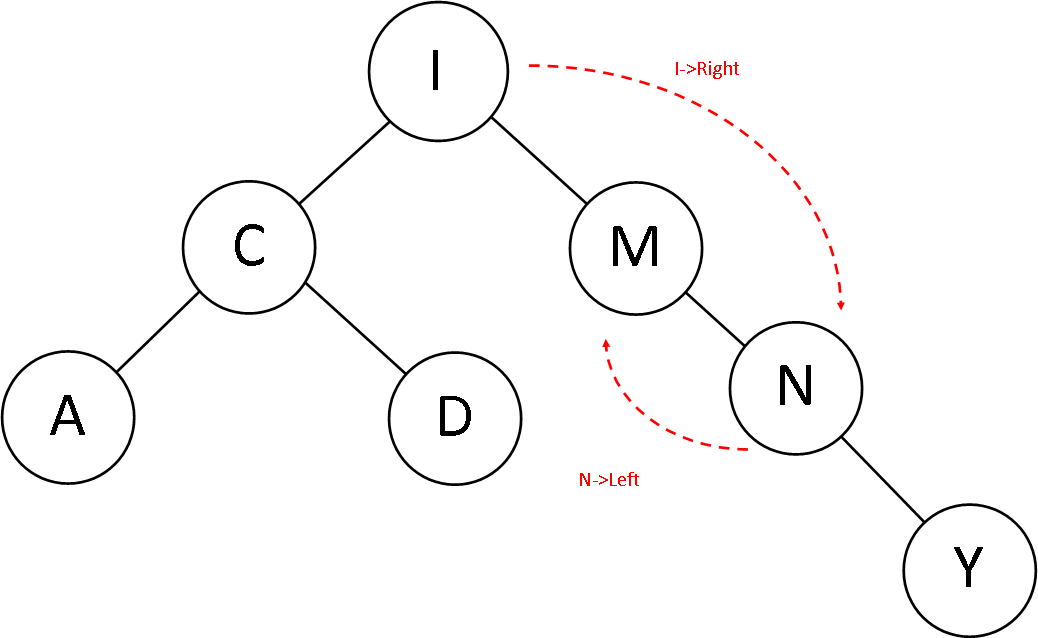
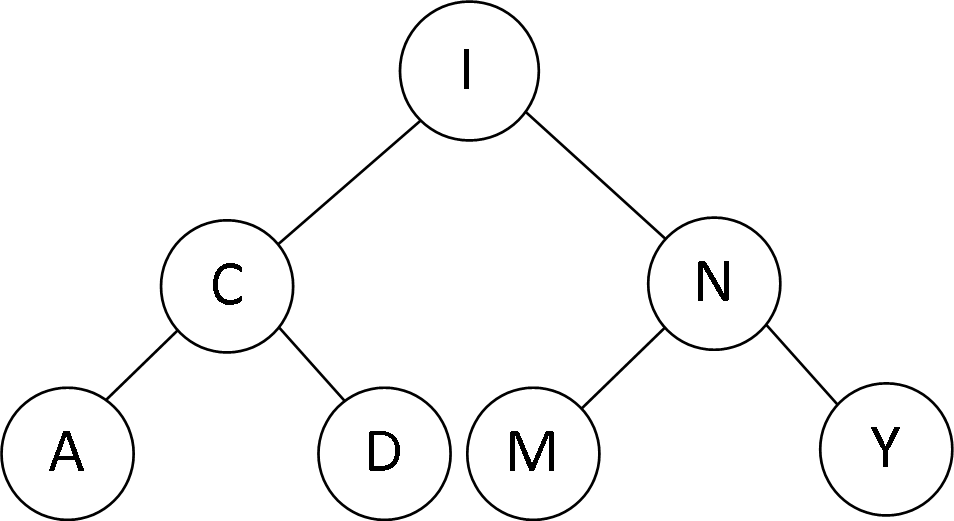


* 1. A **queue** must implement:
     1. **Enqueue:** Add an element to the end of the queue
     2. **Dequeue:** Remove an element from the start of the queue
  2. To ensure these operations run in O(n) time in a singly linked list, we would have the queue keep pointers to the start and end of the queue.
  3. Building a 2,3 Tree with the characters C,O,M,P,L,E,X,I,T,Y:

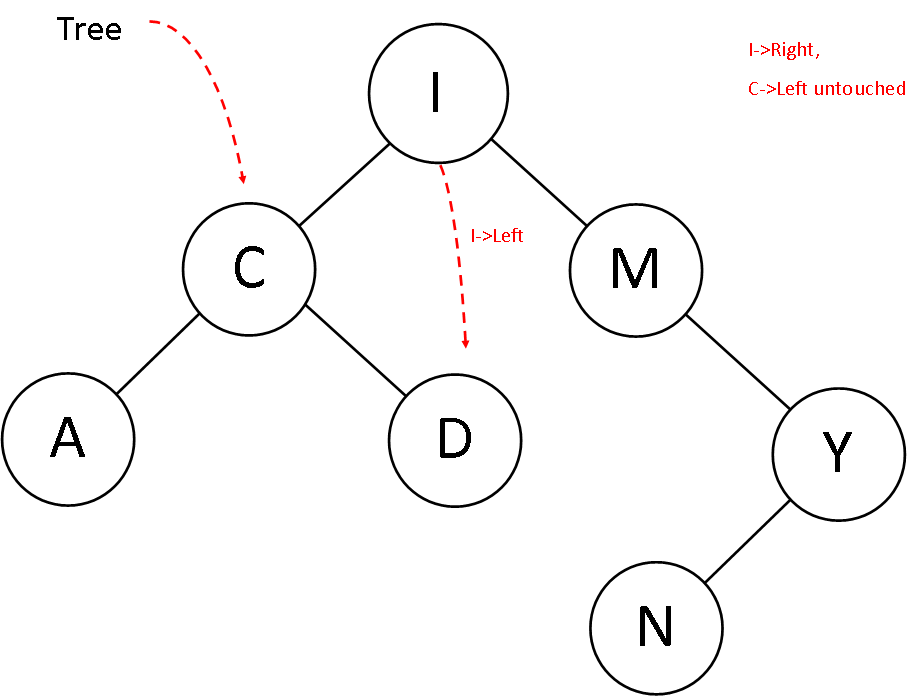
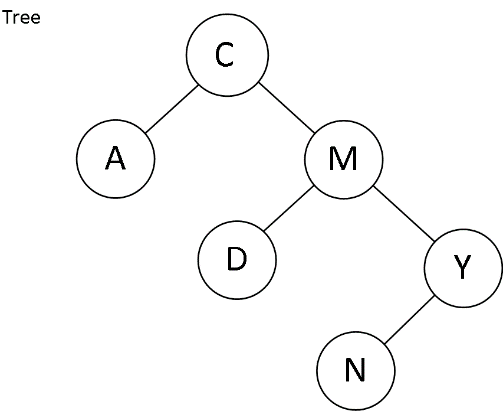


1. Right Rotate on Y, then Left Rotate on N:

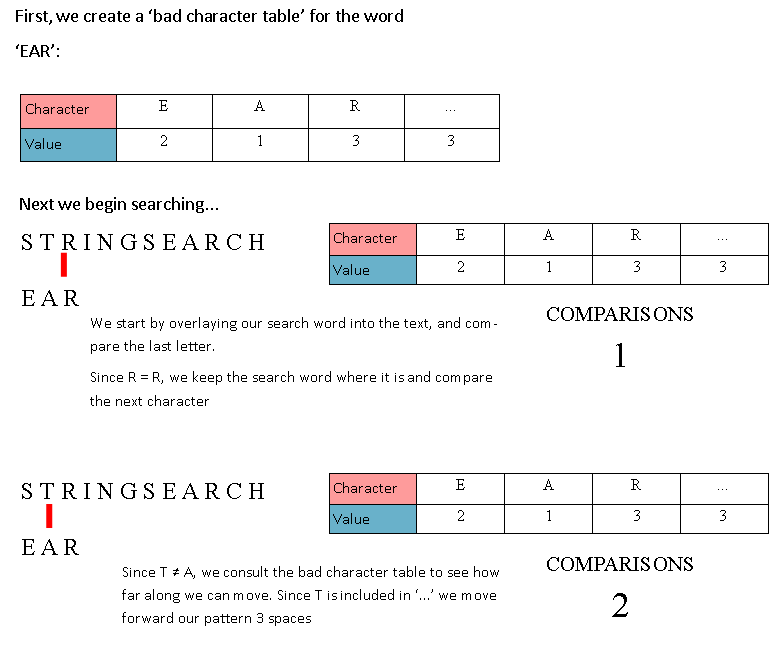


Balanced!! (Although not very well put together)

1. The original in order traversal yields the elements in sorted order: [A,C,D,I,M,N,Y]. Right-rotating the whole tree gives the following tree:



It’s in order traversal yields [A,C,D,M,N,Y], the same as the original result! This is because rotating a binary search tree preserves the principle features of the tree, namely that the in order traversal always yields the contents in ascending order.



1. 