Question 1a (written section)

* Every parent/node can have at most 2 children
* The root value is greater than both children
* Left subtree of each nodes is less than the node values
* Right subtree of each node is greater than the node values

Question 1b (written section)

1. Insert: Ο(n)
2. Delete: Ο(n)
3. Find-next: Ο(n)
4. Find-prev: Ο(n)
5. Find-min: Ο(n)
6. Find-max: Ο(n)

Question 1c (written section)

1. “Just to clarify you want us to make a binary search tree that implements all the given methods recursively, is it also safe to assume that each node will have positive integer values?, Lastly is it safe to assume that we can use any number of parameters for each method?”
2. “for findNextRec() if there does not exist a number that is bigger than the given node, do we return null or the same number? This goes the same for findPrevRec()”
3. InsertRec(1) -> 1 insertRec(2) -> 1 etc… 1

/ \ / \ / \

null null null 2 0 2

findNextRec(1) ->2 findPrevRec(1) -> 0 findMinRec(tree) -> 0

findMaxRec(tree) -> 2

deleteRec(0) -> 1

/ \

null 2

1. Some issues that I do have on space is the implicit space with recursion. All of these algorithms can be done iteratively and not use that extra space
2. Some tradeoffs that we have with the recursive solutions are that they are easier to implement but the implicit space complexity grows rather than not having this problem with iteratively solution.

Question 2a (written section)

[5, 6, 7, 10, 11, 12, 16, 17, 18, 19, 29]

Note: BST values weren’t altered, leading zeros were left out so it can be easier to read

Question 2b (written section)

One method of outputting a sorted list is by traversing the tree inorder = (leftchild, root, rightchild). Since we know every child to the left of the root is less than the given root and same goes for the right of the root. We can print traverse to the left most of the tree and print out the values in that order