**CS2030S RECITATION 09**

Code for trees tend to be recursive.

**Q1**

1. 5 times. ok
2. When the tree is 1 -2 -3 4 -5 . ie a linear tree ok
3. See code? Wrong. Just wrap eagerContains in a lambda.
4. See code? Just keep going to the right until the next right is empty. But need to account for situation when your original tree is empty.

**Q2** (got a few types actually: pre-order, post-order, breadth-first)

Supplier => takes in nothing, gives out something. Ie () -> this.Lazy.of(xxx))

Consumer => takes in something, void action. Ie ( x -> System.out.println(x))

Both C and S: eg mapping because .map( x -> x+1) returns something also

1. M, ok

D,

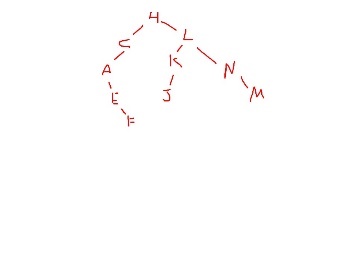
A,

S,

P,

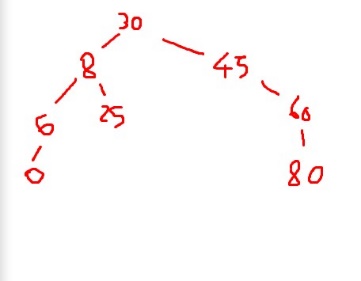
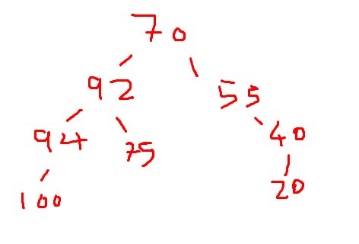
R,

W,

1. 

* E – F branch should be right of C, not right branch of A
* M is left of N, not right of N
* So eg if you replaced K with D, it is not possible because L is already searching through the right subtree, but D would have had to been on the left side thus should have been printed before L. thus not possible

**Q3**

1.  given the code, only 30 is available. Everything else on the left and right are frozen. So it is something like () -> this.leftTree().add(8).add(6) where the add(6) is another () -> this.leftTree().add(6)
2.  clarification: the number 20 is on the right of 40
3. Yes, just that it is inversed ie the left is for values larger than the Node while the right are for values smaller than the node.