Hello everyone. My name is Anna Yeh. For this cs131 project, I'm gonna talk about the tree traversals. In simple words traversing a tree mean to read or process the content of a tree node exactly once in some order. Since tree is not a linear data structure like an array or a linked list where we have a logical start and a logical end, therefore it can be traversed in many ways. We can broadly classify traversing into two ways, the breadth first traversal and the depth first traversal. These techniques are general ways of traversing a graph, and since tree is a special form of graph, we can use them. Breadth first traversal is also known as level order traversal in trees. And depth first has futher categories. It can be Preorder Traversal, Inorder traversal, or Postorder traversal.

For the preorder traversal, you will have to start from the first node, and whenever you reach a node for the first time, you will give it as output, so the output will be A because this is the root, and you will be starting from the root. Then keep moving to the left node, so when you reach B, it will output B. Then you reach D, it will out put D. Wherever our node is having either the left or the right or both the pointers are null like this D is a leaf, so the left and right will be null, and same as E and F. C is having just right as null. After D it comes to B, but B was already visited, so we will go to E. When we reach it, it will print E. After E, we will reach B again, but B was already traversed, and same as A. When we reach C, it will print C, and then it comes to F, so it will print F. After F, we will reach C again, and after C, we will again reach A. So the preorder traversal has been simply calculated to be ABDECF

For the inorder traversal, we also need to start from the root node, and A is the root node, so you will reach A first. In inorder traversal, when you reach the node for the second time, you will have to print it. Since we are reaching A for the first time, we will not print it. Same as B and D. There is no left pointer on D, so we will reach D again. We are reaching D for the second time, so we will print D. After that we will go to the right subtree of D. Since there is no right subtree, we will come back to D. After D, we will reach B, and here B is being reached for the second time, so you will have to print B. After B, you will reach E for the first time, so you shouldn't do anything. There is no left subtree on E, you will need to come back to E. Since you are reaching E for the second time, you will have to print E. After this, you will need to go to the right subtree of E, and since there is no right subtree, you reach E again. After that, you will reach B again, and you will reach A for the second time, so you will need to print A. After that, you will reach to C. C is being reach for the first time, so you shouldn't do anything. After that, you will reach F for the first time, and you shouldn't do anything. There is no left subtree

of F, you will need to come back to F. Since you are reaching F for the second time, you will have to print F. After this, you will need to go to the right subtree of F, and since there is no right subtree, you reach F again. After that, you will reach C for the second time, so you will need to print C. There is no right subtree on C, so you will come back to C, and then come back to A. So the inorder traversal has been simply calculated to be DBEAFC

For postorder traversal, we will start from A again. In postorder traversal, first we will traverse the left subtree, then traverse the right subtree, and then the root node. Starting from A, the first time you will not do anything, same as B and D. D is reached for the first time, then you will go to the left subtree of D, but there is no left subtree, so you will come back to D, and this is for the second time. Then go to the right subtree of D, but there is no right subtree, so you will reach D again. This is the third time we reached D, so it will print D. After that, you will reach to B, and B is reach for the second time, so you shouldn't do anything. Then you will go to E for the first time, so you will not do anything. Then you will go to its left subtree, and then come back to E, and then go to its right subtree. There is no right subtree, so you will come back to E. Since left and right subtree of E have been traversed, you can print E. Then you will reach B again, and the left and the right subtree of B have all been traversed, and we are reaching B for the third time, so you will print B. After that, you will reach A, and A is being reached for the second time, so you will not do anything. C is being reached for the first time, so you shouldn't do anything. F is also reached for the first time, then you will go to the left subtree of F, but there is no left subtree, so you will come back to F, and this is for the second time. Then go to the right subtree of F, but there is no right subtree, so you will reach F again. This is the third time we reached F, so it will print F. After that, you will reach C, and C is reached for the second time, so you shouldn't do anything. Then go to the right subtree of C, but there is no right subtree, so you will reach C again. This is the third time we reached C, so it will print C. When you reach A, this the third time we reached A, so it will print A. So the postorder traversal has been simply calculated to be DEBFCA.