For my design, I used a recursive Depth First Search algorithm to count subtrees with an even number of nodes. First, I made a function called depthFirstSearch() which consumed the tree, a root node, a boolean array which , and an address to an int to keep track of how many even subtrees could be made. Then I called the depthFirstSearch in another function called maxEdgesRemovable which actually returned the max edges removable.

For the depthFirstSearch(), I initially set the current node to true in its boolean array so that the node would not be double checked during the recursive function. Then I created an int and initialized it to 0 to keep track of how many nodes are in the current subtree that the function is checking. The function then enters a for loop that loops depending on how many children the current node has, if the node has no children, the function skips the loop and hits its base case and returns 1. If the node does have children, int v is set to a child and then its correlating position in the boolean array is checked to see if the node has already been checked or not. If it has not, I make a recursive call and store the number of nodes in the current subtree into the int subtreeNodeCount. Then I check if the subTree has an even number of nodes, if it does I increase the answer int by 1, else I add the number of nodes in the subTree to the existing number of nodes to keep track of the currentTree. The function then returns the numNodes + 1 in order to count the current node as well.

For the maxEdgesRemovable(), I created a boolean array called visit which is the size of the tree in order to keep track of which nodes have been checked during the depthFirstSearch(). Next, I set all the elements in the boolean array to be false and then declare an int called ans that will be used to store the final answer of edges that we can remove. Then I call the depthFirstSearch() with the consumed tree, 0 (which is the root node), the array visit, and ans as an address so it can be modified in the function. Lastly, I return the answer.