Exercise 3:

Time complexity of genData(): is O(n) where n is the size of the input, because in this function in each iteration we are trying to generate a random numbers and push it in to the vector and the number of iteration is the size of the vector which is equal to the number of inputs (the time complexity of the operations are done in each iteration is constant ), so O(n).

Time complexity of makeBST() : insert() function is called inside the makeBST() function, which has a time complexity of O(logn) on average when the tree is balanced (because we are kinda inserting the nodes in a sorted way, smaller value on left, bigger value on right, so it is O(log)), but in the worst case scenario, the time complexity of insert() function is O(n) when the tree is very unbalanced. This insert() function is called in a for loop for n number of iterations which n is the size of the vector (inputs), so the time complexity of makeBST on average is O(nlogn) if the tree is balanced and in the worst case the time complexity of makeBST() is O(n^2), or more specifically O(n height ), in the worst case the time complexity of insert function in fact is based on the height of the tree

Time complexity of printBT() : printBT() is calling inorder() function from the header class . the time compelexity of inorder traversal for a binary tree is O(n), where n is the size of the tree (number of inputs here). The same thing is true about preorder traversal.

Time complexity of height() : if the tree is balanced the time complexity is O(logn), but in the worst case where the tree is not balanced, is O(n) where n is height of the tree