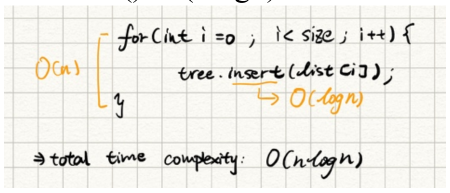
Assignment 3

Exercise 4:

Time Complexity:
a.
genData(): O(n)

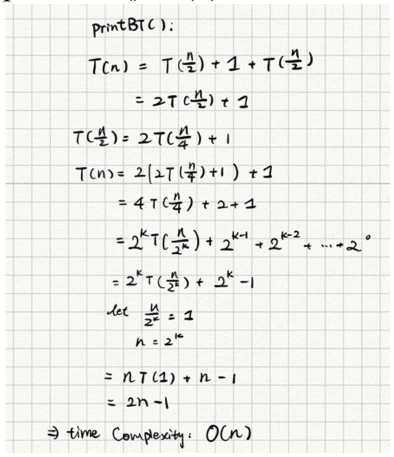
makeBST(): O(nlogn)



```
        for(int i =0 ;  i< size ; i++){
O(n)            tree.insert(list[i]);
                      ↳ O(logn)
        }
```

⇒ total time complexity: O(nlogn)

printBST(): O(n)



printBT():

$$T(n) = T(\tfrac{n}{2}) + 1 + T(\tfrac{n}{2})$$
$$= 2T(\tfrac{n}{2}) + 1$$
$$T(\tfrac{n}{2}) = 2T(\tfrac{n}{4}) + 1$$
$$T(n) = 2(2T(\tfrac{n}{4})+1) + 1$$
$$= 4T(\tfrac{n}{4}) + 2 + 1$$
$$= 2^k T(\tfrac{n}{2^k}) + 2^{k-1} + 2^{k-2} + \dots + 2^0$$
$$= 2^k T(\tfrac{n}{2^k}) + 2^k - 1$$

let $\frac{n}{2^k} = 1$

$n = 2^k$

$$= nT(1) + n - 1$$
$$= 2n - 1$$

⇒ time Complexity: O(n)

height(): O(1) since height is a data member in my BST class

remove(): O(logn)

1. Takes O(logn) to find the target element
2. O(1) to delete the target node
3. O(1) for rotation operations to happen

mergeBST(): two BSTs: A and B where m is the size of A, and n is the size of B
O(nlogn)

1. Takes O(m) to copy all the nodes from A to a new tree C
2. Takes O(nlogn) to insert all the node from B to C

InfixPostfixExpression(): n is the length of the string: O(n)
Scan the expression from left to right, and put them into an expression tree structure

infixExprTree(): it basically traverses the tree in a post order fashion, so it is: O(n)

Assignment 3

b.

Space Complexity:
mergeBST(): two BSTs: A and B where m is the size of A, and n is the size of B:
we need m+n new nodes, so the space complexity is O(m+n)