Name: Md Habibur Rony
Student ID: 984582
Weekday: Week 1- Day 4

Answer to the Q. No. R-2.8:
Step 1: **22**, 15, 26, 44, 10, **3**, 9, 13, 29, 25
Step 2: 3, **15**, 26, 44, 10, 22, **9**, 13, 29, 25
Step 3: 3, 9, **26**, 44, **10**, 22, 15, 13, 29, 25
Step 4: 3, 9, 10, **44**, 26, 22, 15, **13**, 29, 25
Step 5: 3, 9, 10, 13, **26**, 22, **15**, 44, 29, 25
Step 6: 3, 9, 10, 13, 15, **22**, 26, 44, 29, 25
Step 7: 3, 9, 10, 13, 15, 22, 26, 44, 29, 25
Step 8: 3, 9, 10, 13, 15, 22, 25, 44, 29, 26
Step 9: 3, 9, 10, 13, 15, 22, 25, 26 29, 44
Step 10: 3, 9, 10, 13, 15, 22, 25, 26 29, 44

Selection sort finds minimum element and swap it with the first element of unsorted section. The performance of the selection sort for insertion of each element is O(1), so for all elements take O(n) times.
The performance of selection sort for all elements will be $O(n^2)$. Because it traverse to all the elements to find the minimum value.

Answer to the Q. No. R-2.9:
Step 1: 22, 15, 26, 44, 10, 3, 9, 13, 29, 25
Step 2: 3, 15, 26, 44, 10, 22, 9, 13, 29, 25
Step 3: 3, 9, 26, 44, 10, 22, 15, 13, 29, 25
Step 4: 3, 9, 10, 44, 26, 22, 15, 13, 29, 25
Step 5: 3, 9, 10, 13, 26, 22, **15**, 44, 29, 25
Step 6: 3, 9, 10, 13, 15, 22, 26, 44, 29, 25
Step 7: 3, 9, 10, 13, 15, 22, 26, 44, 29, 25
Step 8: 3, 9, 10, 13, 15, 22, 25, 44, 29, 26
Step 9: 3, 9, 10, 13, 15, 22, 25, 26, 29, 44
Step 10: 3, 9, 10, 13, 15, 22, 25, 26, 29, 44

In the insertion sort, we take an unsorted element and swap it to left until it find a proper position, which takes O(n) time. So, for n elements, it takes $O(n^2)$ time.

Answer to the Q. No. R-2.10:

For insertion sort worst case will be the sequence of elements in which it is sorted by opposite order. For example: **5 4 3 2 1**

Above list will take $O(n^2)$ time.

Answer to the Q. No. R-2.13:

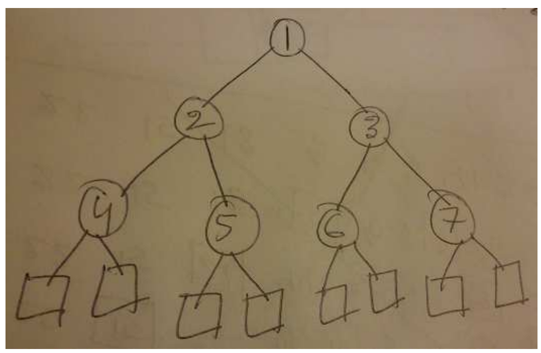
Yes. The tree T is a heap.

Justification:

Suppose Vector S which consists of n element with sorted order from index 1 then we will get -

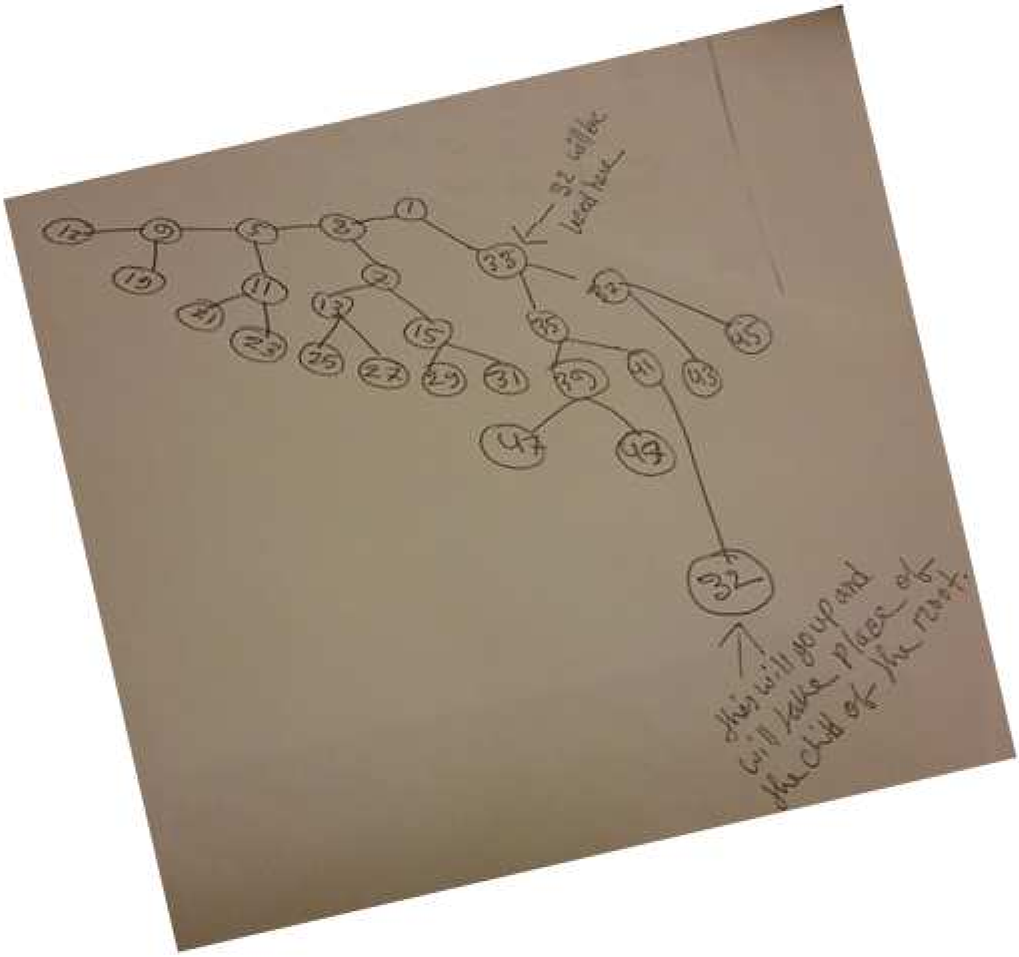
Say, n = 7

	1	2	3	4	5	6	7			
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[etc]



We know for heap order, for every internal node v other than the root, key(v) >= key(parent(v)). That means child will be always greater than parent. So a binary tree represented in a sorted order vector S, we can say it's a heap because every child is greater than or equal to parent.

Answer to the Q. No. R-2.18:



Answer to the Q. No. C-2.32:

Algorithm GetSmaller(T, node, x)
Input: Tree T, Position node and value x
Output: list of the smaller sequence.

```
if node = empty V node.element > x then
        return list
else if node.element <= x then
        list.add(node.element)
GetSmaller(T, node.left, x)
GetSmaller(T, node.right,x)
```