Launching into Computer Science Assignment 1 : Part 1

Question 1: Suppose you are doing a sequential search of the list [15, 18, 2, 19, 18, 0, 8, 14, 19, 14]. How many comparisons would you need to do in order to find the key 18? Show the steps.

Comparison 1:

Compare 18 with index 0 (15).

15 != 18.

Move on to next index.

Comparison 2:

Compare 18 with index 1 (18).

18 == 18.

Stop.

Two comparisons are needed to find 18.

Question 2: Suppose you have following list of numbers to sort [19, 1, 9, 7, 3, 10, 13, 15, 8, 12]. Show the partially sorted list after three complete phases of bubble sort.

After 3 complete phases of a bubble sort, the partially sorted list would be [1, 3, 7, 9, 10, 8, 12, 13, 15, 19]. The complete step by step of each phase is given below.

Phase 1:

[19, 1, 9, 7, 3, 10, 13, 15, 8, 12] → [1, 19, 9, 7, 3, 10, 13, 15, 8, 12]

[1, 19, 9, 7, 3, 10, 13, 15, 8, 12] → [1, 9, 19, 7, 3, 10, 13, 15, 8, 12]

[1, 9, 19, 7, 3, 10, 13, 15, 8, 12] → [1, 9, 7, 19, 3, 10, 13, 15, 8, 12]

[1, 9, 7, 19, 3, 10, 13, 15, 8, 12] → [1, 9, 7, 3, 19, 10, 13, 15, 8, 12]

[1, 9, 7, 3, 19, 10, 13, 15, 8, 12] → [1, 9, 7, 3, 10, 19, 13, 15, 8, 12]

[1, 9, 7, 3, 10, 19, 13, 15, 8, 12] → [1, 9, 7, 3, 10, 13, 19, 15, 8, 12]

[1, 9, 7, 3, 10, 13, 19, 15, 8, 12] → [1, 9, 7, 3, 10, 13, 15, 19, 8, 12]

[1, 9, 7, 3, 10, 13, 15, 19, 8, 12] → [1, 9, 7, 3, 10, 13, 15, 8, 19, 12]

[1, 9, 7, 3, 10, 13, 15, 8, 19, 12] → [1, 9, 7, 3, 10, 13, 15, 8, 12, 19]

Phase 2:

[1, 9, 7, 3, 10, 13, 15, 8, 12, 19] → [1, 9, 7, 3, 10, 13, 15, 8, 12, 19]

[1, 9, 7, 3, 10, 13, 15, 8, 12, 19] → [1, 7, 9, 3, 10, 13, 15, 8, 12, 19]

[1, 7, 9, 3, 10, 13, 15, 8, 12, 19] → [1, 7, 3, 9, 10, 13, 15, 8, 12, 19]

[1, 7, 3, 9, 10, 13, 15, 8, 12, 19] → [1, 7, 3, 9, 10, 13, 15, 8, 12, 19]

[1, 7, 3, 9, 10, 13, 15, 8, 12, 19] → [1, 7, 3, 9, 10, 13, 15, 8, 12, 19]

[1, 7, 3, 9, 10, 13, 15, 8, 12, 19] → [1, 7, 3, 9, 10, 13, 15, 8, 12, 19]

[1, 7, 3, 9, 10, 13, 15, 8, 12, 19] → [1, 7, 3, 9, 10, 13, 8, 15, 12, 19]

[1, 7, 3, 9, 10, 13, 8, 15, 12, 19] → [1, 7, 3, 9, 10, 13, 8, 12, 15, 19]

[1, 7, 3, 9, 10, 13, 8, 12, 15, 19] → [1, 7, 3, 9, 10, 13, 8, 12, 15, 19]

Phase 3:

[1, 7, 3, 9, 10, 13, 8, 12, 15, 19] → [1, 7, 3, 9, 10, 13, 8, 12, 15, 19]

[1, 7, 3, 9, 10, 13, 8, 12, 15, 19] → [1, 3, 7, 9, 10, 13, 8, 12, 15, 19]

[1, 3, 7, 9, 10, 13, 8, 12, 15, 19] → [1, 3, 7, 9, 10, 13, 8, 12, 15, 19]

[1, 3, 7, 9, 10, 13, 8, 12, 15, 19] → [1, 3, 7, 9, 10, 13, 8, 12, 15, 19]

[1, 3, 7, 9, 10, 13, 8, 12, 15, 19] → [1, 3, 7, 9, 10, 13, 8, 12, 15, 19]

[1, 3, 7, 9, 10, 13, 8, 12, 15, 19] → [1, 3, 7, 9, 10, 8, 13, 12, 15, 19]

[1, 3, 7, 9, 10, 8, 13, 12, 15, 19] → [1, 3, 7, 9, 10, 8, 12, 13, 15, 19]

[1, 3, 7, 9, 10, 8, 12, 13, 15, 19] → [1, 3, 7, 9, 10, 8, 12, 13, 15, 19]

[1, 3, 7, 9, 10, 8, 12, 13, 15, 19] → [1, 3, 7, 9, 10, 8, 12, 13, 15, 19]

Question 3: Given the statement below

```
x = BinaryTree('a')
insert_left(x,'b') insert_right(x,'c')
insert_right(get_right_child(x),'d')
insert_left(get_right_child(get_right_child(x)),'e')
```

Which of these answers is the correct representation of the tree? Show your working out.

A. `['a', ['b', [], []], ['c', [], ['d', [], []]]]`

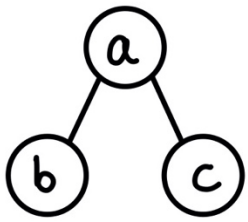
B. `['a', ['c', [], ['d', ['e', [], []], []]], ['b', [], []]]`

C. `['a', ['b', [], []], ['c', [], ['d', ['e', [], []], []]]]`

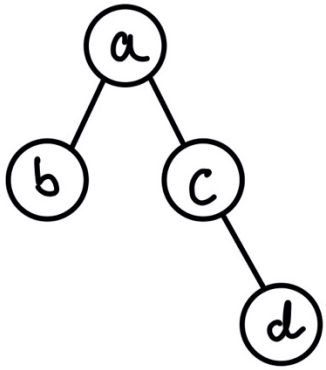
D. `['a', ['b', [], ['d', ['e', [], []], []]], ['c', [], []]]`

`x = BinaryTree('a')` gives `['a', [], []]`.

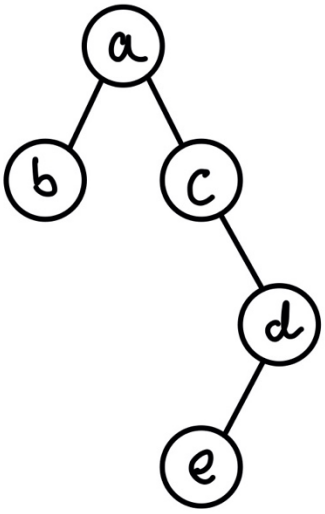
insert_left(x,'b') insert_right(x,'c') gives [‘a’, [‘b’, [], []], [‘c’, [], []]].



insert_right(get_right_child(x),'d') gives [‘a’, [‘b’, [], []], [‘c’, [], [‘d’, [], []]]].



`insert_left(get_right_child(get_right_child(x)),'e')` gives `[‘a’, [‘b’, [], []], [‘c’, [], [‘d’, [‘e’, [], []], []]]]`.

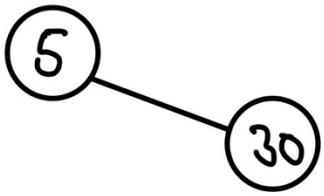


This statement produces answer C.

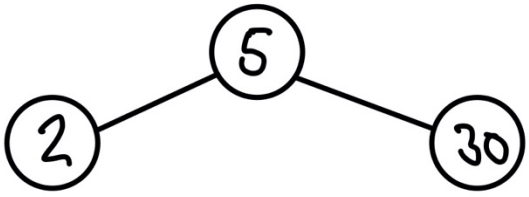
Question 4: Draw a tree showing a correct binary search tree given that the keys were inserted in the following order 5, 30, 2, 40, 25, 4.

Step 1: Make 5 the initial node.

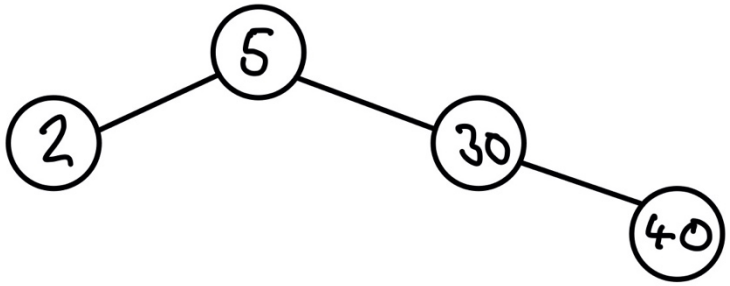
Step 2: 30 > 5 so it becomes the right child of 5.



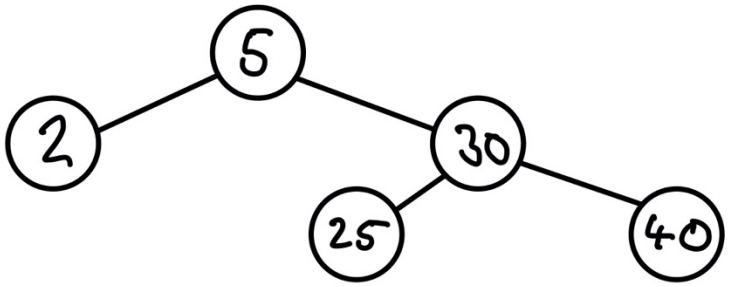
Step 3: 2 < 5 so it becomes the left child of 5.



Step 4: 40 > 5 so move down the right edge to 30. Then 40 > 30 so it becomes the right child of 30.



Step 5: 25 > 5 so move down the right edge to 30. 25 < 30 so it becomes the left child of 30.



Step 6: 4 < 5 so move down the left edge to 2. 4 > 2 so it becomes the right child of 2.