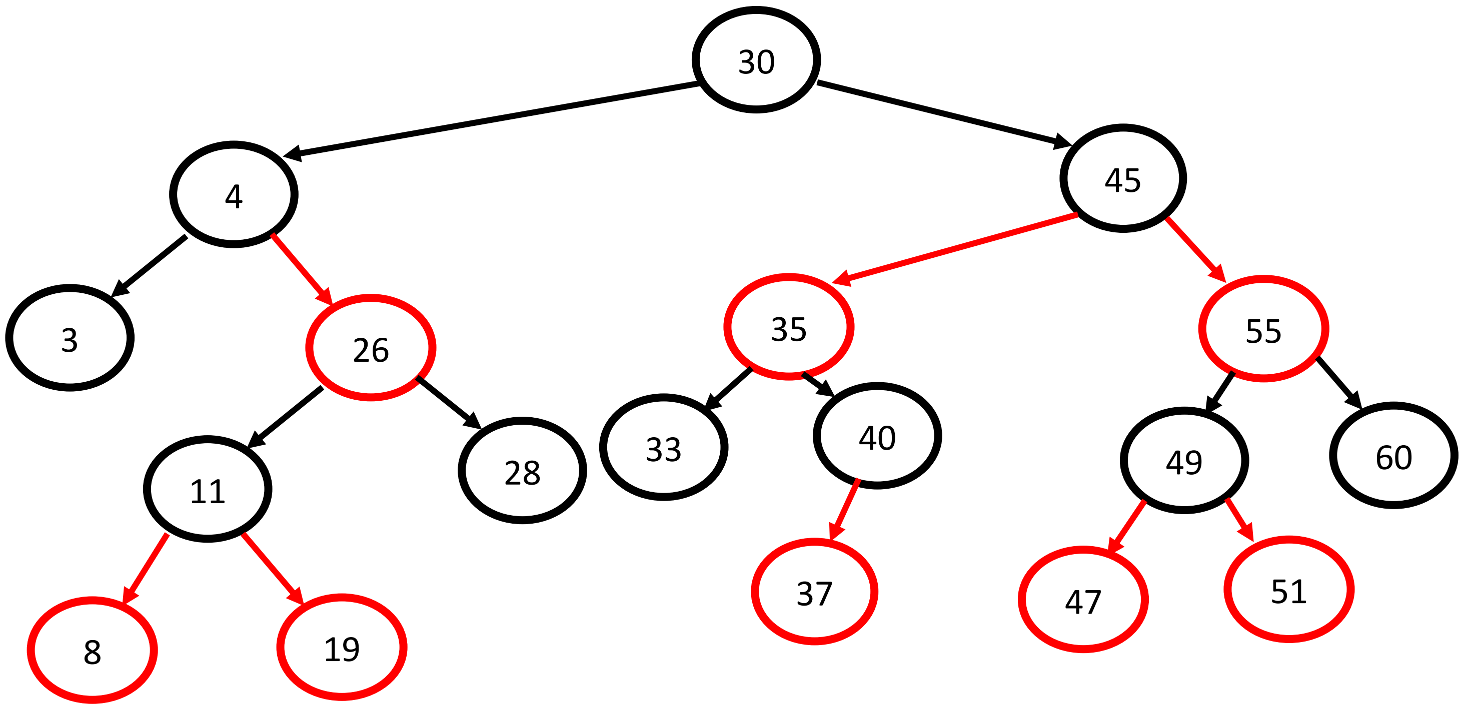
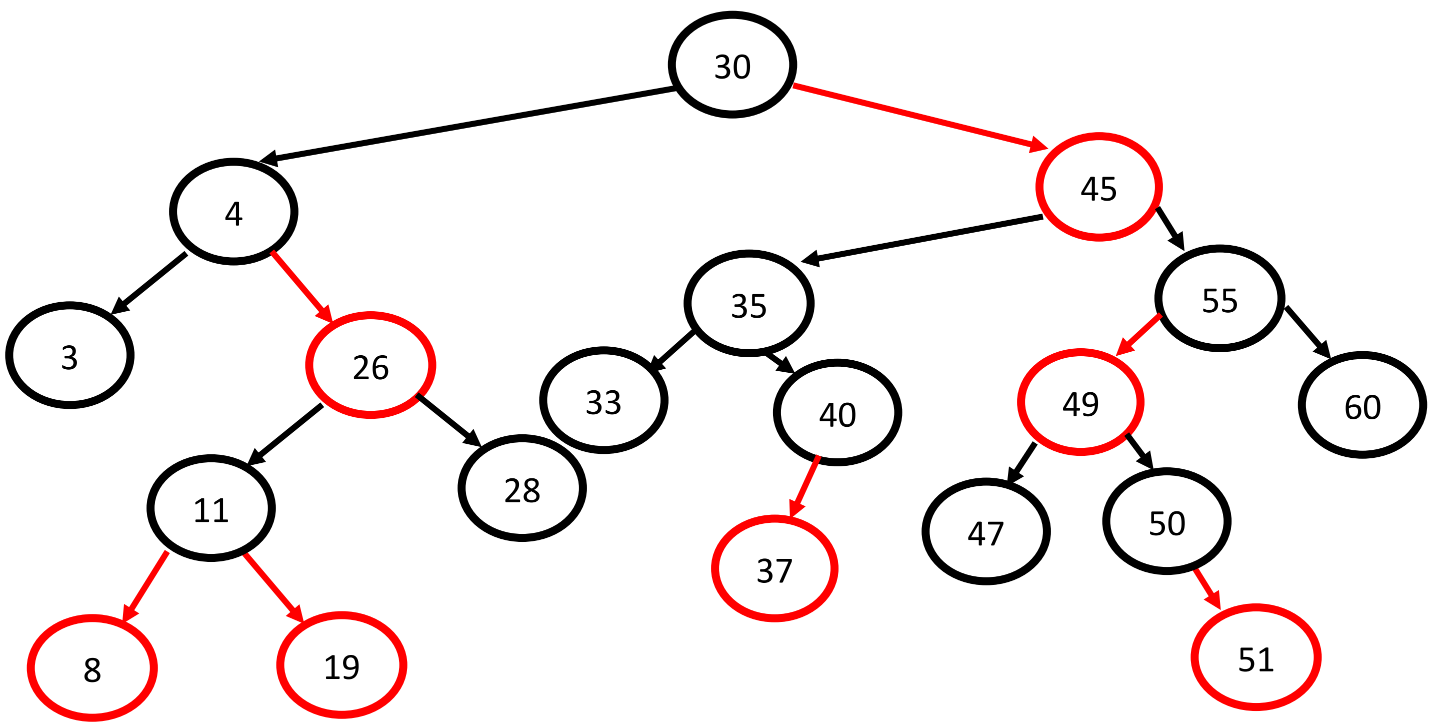
Murat Sevinç
21702603
Section 1

Question 1:

Part a; Equivalent Red-Black Tree:



Part b; After inserting 50 Red-Black Tree:



Question 2:

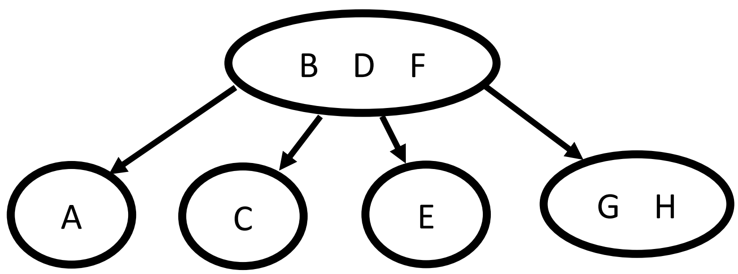
Part a)

$3^h - 1$

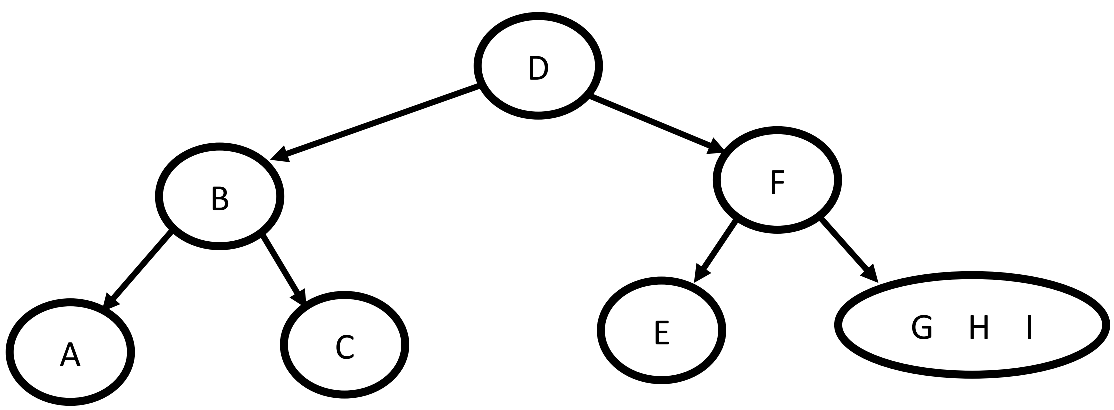
Part b)

Letter "i" makes the 2-3-4 tree to grow to height 3 for the first time.

Before inserting "i":



After inserting "i":



Part c)

O (N). Since red-black trees are built in a sorted manner, by simply applying an in-order traversal we can obtain a sorted list.

Part d)

No. A red-black trees must start with a black root, yet, the root of a subtree of a red-black tree can be red, which is not allowed.

Question 3:

Part a)

In this problem, we will need to create a hash table by using the **keys** of the elements in the array (if first element of the array is 5 for example, in the hash table, 5th element will contain the location of 5, which is 0 in this case). Algorithm will perform this for every element and it will be spending O(N) time.

After the initialization of the hash table, again for every element in the array, the algorithm will **assume as the first element is correct** and it will look up for the **second element**, if it finds, it will return these two numbers. It will do this by first **subtracting the current element from the target value**. The remaining number is the second correct element. If, in the hash table, that element exists, then it will get the content of the remaining element and **this will be the second key index**.

The main purpose of using hash tables in this example is to reduce the time for the search time. By using hash tables, this time reduces from O(N) to **O(1)**.

Part b)

When collisions solved by linear probing:

Slot	0	1	2	3	4	5	6
Content	30	15	22	11	14	11	

When collisions solved by quadratic probing:

Slot	0	1	2	3	4	5	6
Content	30	15	22	11	14		11