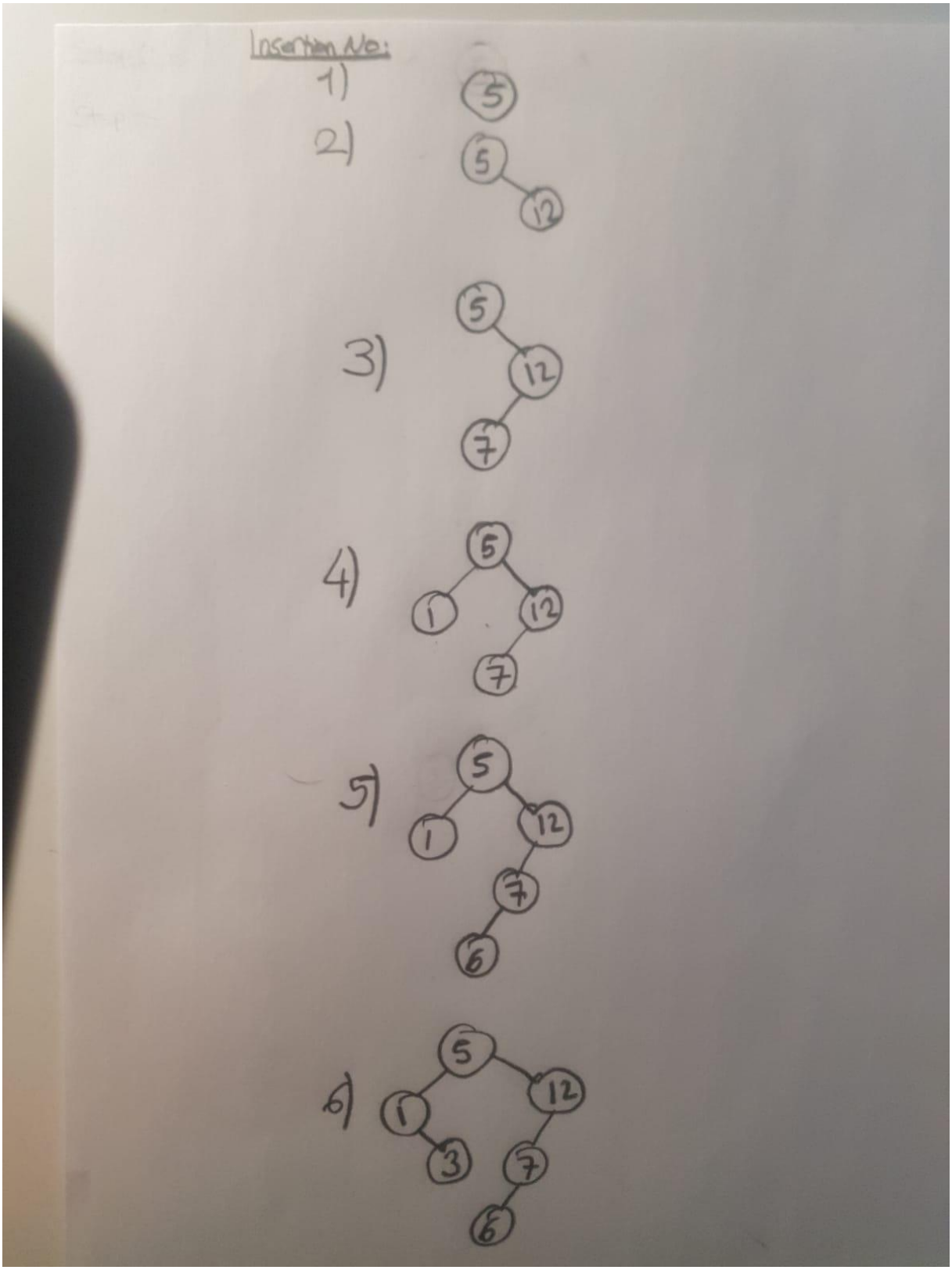
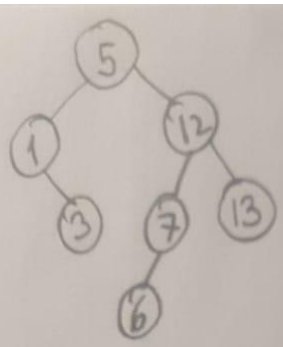
Name: Mehmet Bora Kurucu ID: 21703404 Section: 2

Q1A)

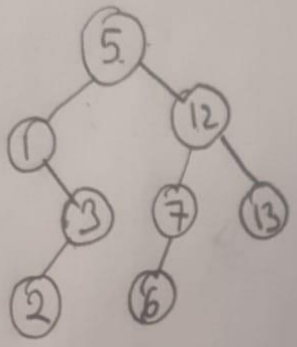


Insertando

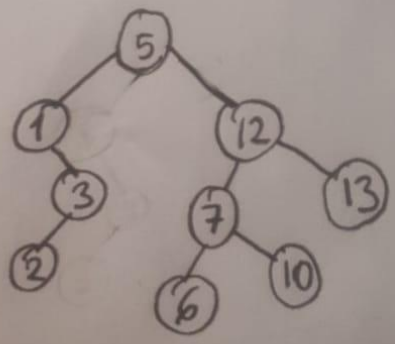
7)



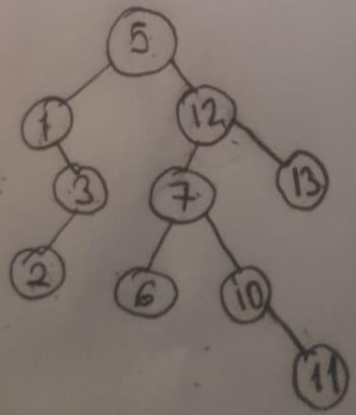
8)



9)



10)



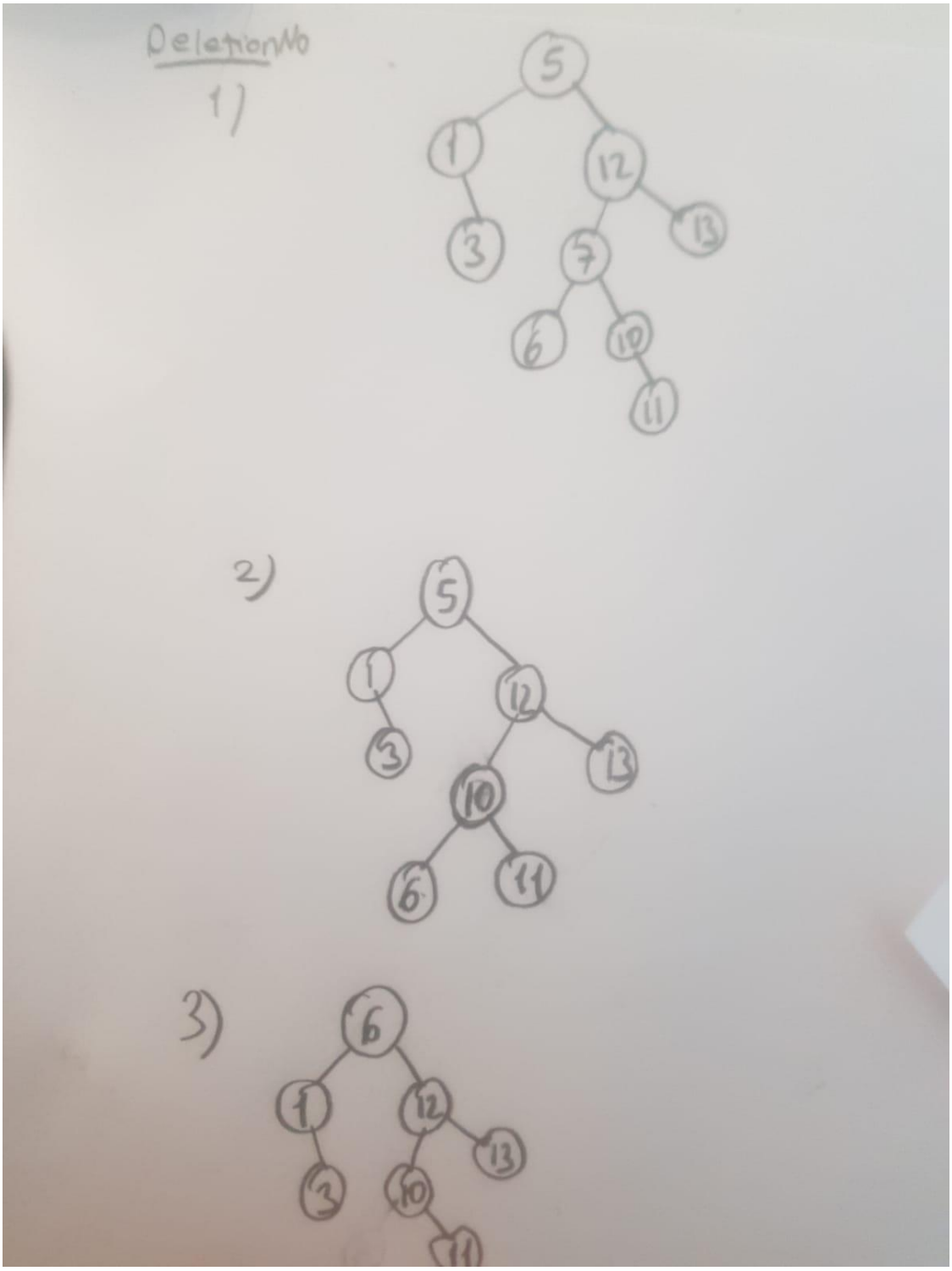
Q1B)

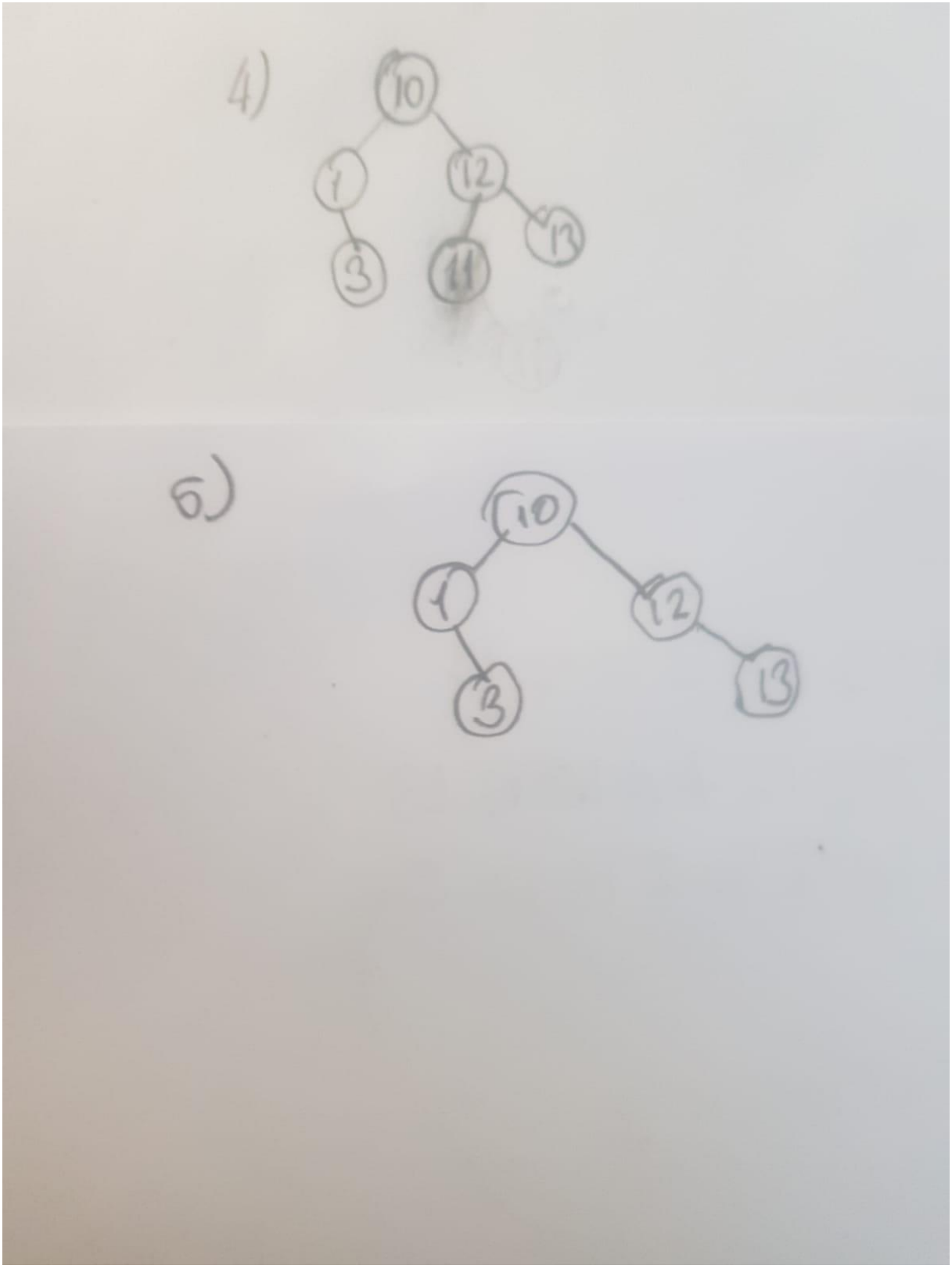
Inorder Traversal: 1 2 3 5 6 7 10 11 12 13

Preorder Traversal: 5 1 3 2 12 7 6 10 11 13

Postorder Traversal: 2 3 1 6 11 10 7 13 12 5

Q1C)





Q3)

insertItem(int key):

Calls void insertItem(BSTNode *& root,int key). Which finds proper place with recursion and inserts the item.

Average Time Complexity is when the function is considered as T(n)=T(n/2)+1. This is when it goes to one of the sides for most of the times. This can be represented as T(n) = T(n/2^k) + k ,therefore T(n) is O(logn).

Worst Time Complexity occurs when the list is linear. It becomes similar with the case of arrays it behaves like a linearly organized data structure. It scans the items one by one to find the target. Therefore the worst case becomes O(n).

deleteItem(int key):

Calls void deleteItem(BSTNode *& root,int key). Which finds proper place with recursion and deletes the item according to it's children number.

Average Time Complexity is when the function is considered as T(n)=T(n/2)+1. This is when it goes to one of the sides for most of the times. This can be represented as T(n) = T(n/2^k) + k, therefore T(n) is O(logn).

Worst Time Complexity occurs when the list is linear. It becomes similar with the case of arrays it behaves like a linearly organized data structure. It scans the items one by one to find the target. Therefore the worst case becomes O(n).

retrieveItem(int key);

Calls void retrieveItem(BSTNode *& root,int key). Which finds proper place with recursion and retrieves the item.

Average Time Complexity is when the function is considered as T(n)=T(n/2)+1. This is when it goes to one of the sides for most of the times. T(n) = T(n/2^k) + k than T(n) is O(logn).

Worst Time Complexity occurs when the list is linear. It becomes similar with the case of arrays it behaves like a linearly organized data structure. It scans the items one by one to find the target.Therefore the worst case becomes O(n).

inorderTraversal(int& length);

Calls void inorderTraversal(BSTNode * root,int& length,int *&ar,int &index) which does inorder traversal, simultaneously increases index and puts the values into array.

It is a traversal, it has to visit every node one time. So both Average Time Complexity and Worst Time Complexity is O(n).

bool containsSequence(int* seq, int length);

Calls containsSequence(BSTNode * root,int* arr, int length) which find lowest common ancestor of array's last element and the tree This takes O(logn) in average(it goes to one of the subtrees), worst is O(n) (if tree is linearly shaped).

After finding it, it calls traverseFromStart(BSTNode * root,int * arr,int &length,int &track) which checks if the array is a inorder subtree. This takes O(logn) in average(it goes to one of the

subtrees), worst is O(n) (if tree is linearly shaped).To conclude Average Time Complexity of ContainsSequence is O(logn) and the Worst is O(n).

int countNodesDeeperThan(int level);

It calls void countNodesDeeperThan(BSTNode * root,int level,int &count,int start), which makes a level order traversal until it reaches to the desired level, after that it does inorder traversal.

if level is dependent on n,for instance n/2,3n,n:

Time complexity of the level order traversal would be O(n^2) in both average of worst. Since they are O(n) for inorder traversal, it would be O(n^2) for both average and wost case.

Else:

Time complexity of level order is O(1). So only inorder traversal is considered. Then both Average and Worst Time complexities are O(n).

int maxBalancedHeight();

This function calls int calcHeight(BSTNode * root) which makes a postorder traversal, and decides the height according to the heights of it's left and right subtrees. Since it is a traversal it visits every node, both average and worst case time complexities are O(n).

int findLength(BSTNode * root);

Finds number of items by traversing the tree. Since it is a traversal it visits every node, both average and worst case time complexities are O(n).

void printAr(int * arr,int length);

Prints the array. It is used to print the inorder traversal. If length is dependent on tree's size, in my usage it is, both tree and the array has n elements so both average and worst case time complexities are O(n).