Written Part of HW3

Name: Shikhar Bakhda
UNI: ssb2189
Date: 24/3/2017

Q1

Let the proposition that the number of full nodes, nF, and the number of leaves, nL are related by

nF + 1 = nL

be P

1. To prove that P holds true for nF = 1

When there is only one full node, it will be the root. The root must have 2 leaves as full nodes have 2 children. When nF = 1, the children are also leaves, so P holds true for nF = 1

2. To assume that P holds true for nF = k for k > 0

k + 1 = nLk

3. To prove that P holds true for nF = k + 1 for all k > 0

k + 2 = nLk2

We know that the total edges in a tree are n - 1. Leaves have no edges, full nodes have 2 edges, and half nodes (nH) have 1 edge. The total number of nodes is therefore:

n = nF + nL + nH and the total number of edges is:

2nF + nH = n - 1

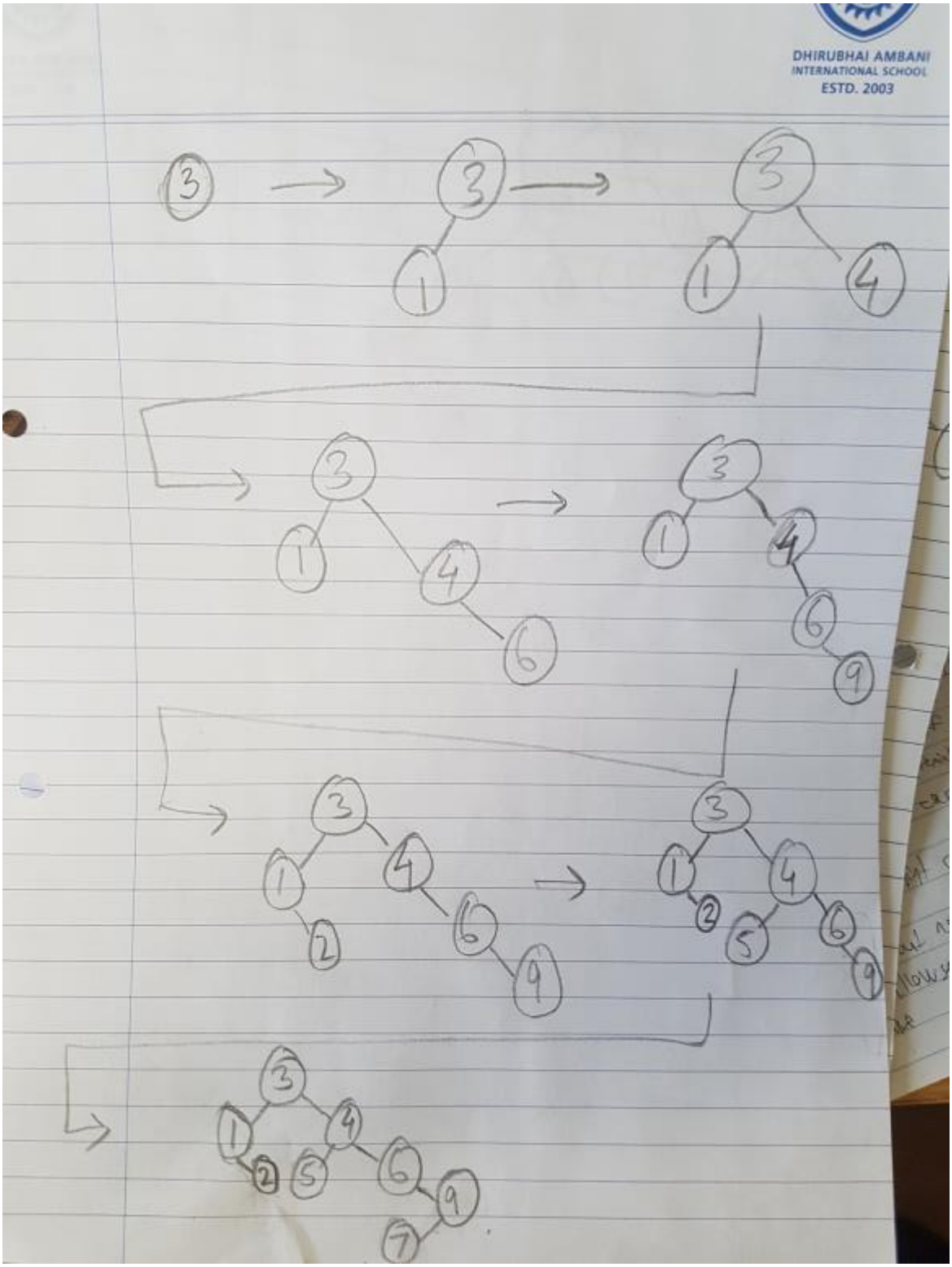
Substituting for nH,
2nF + n - nF - nL = n - 1
nF - NL = -1
nF + 1 = nL

nLk = nF + 1 = k + 1 and
nLk2 = nF + 1 = k + 1 + 1 = k + 2

4.Thus, since P holds true for nF = 1, nF = k, and nF = k + 1, P holds true for all nF > 0.

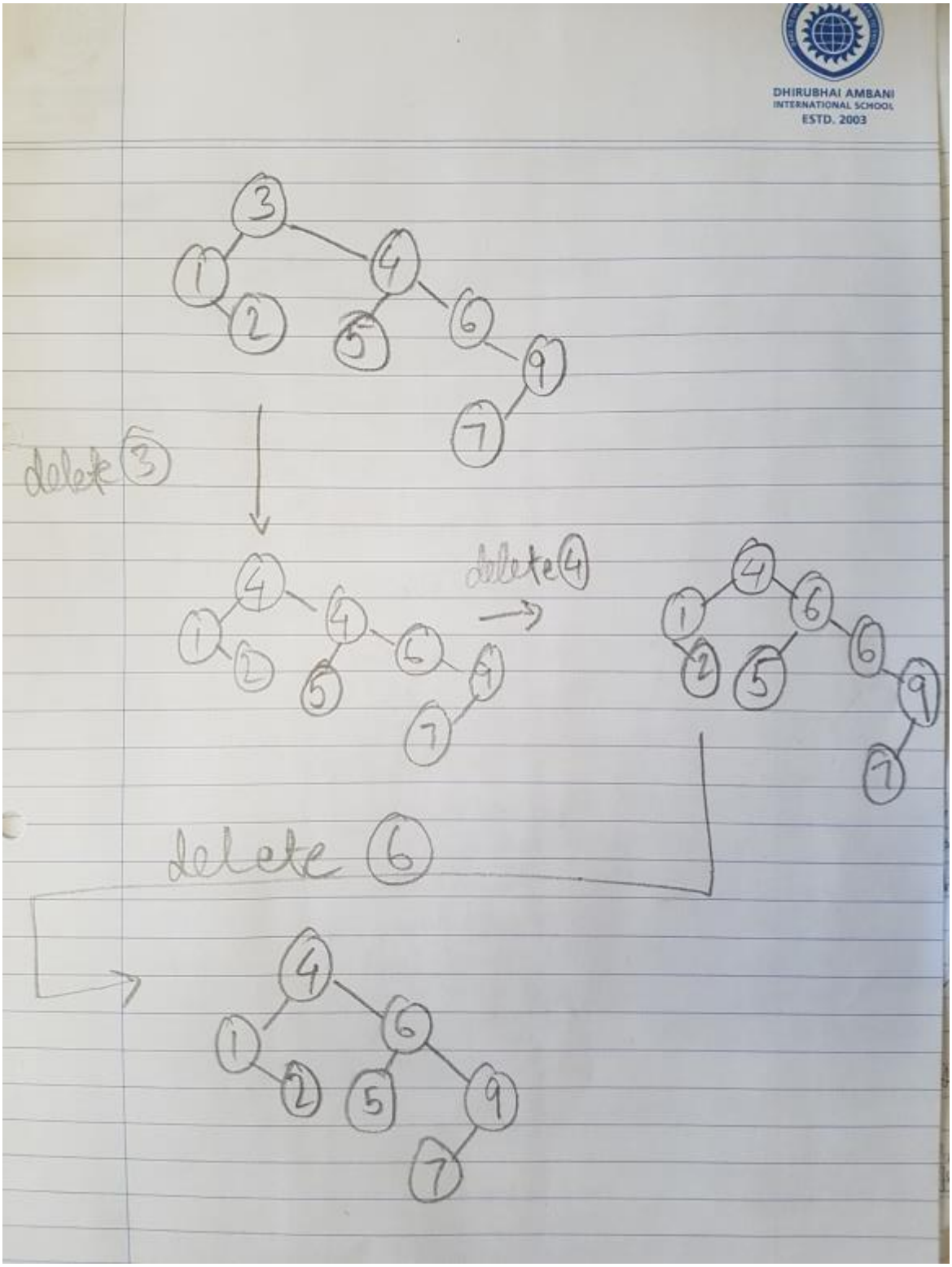
Q2a

Insertion



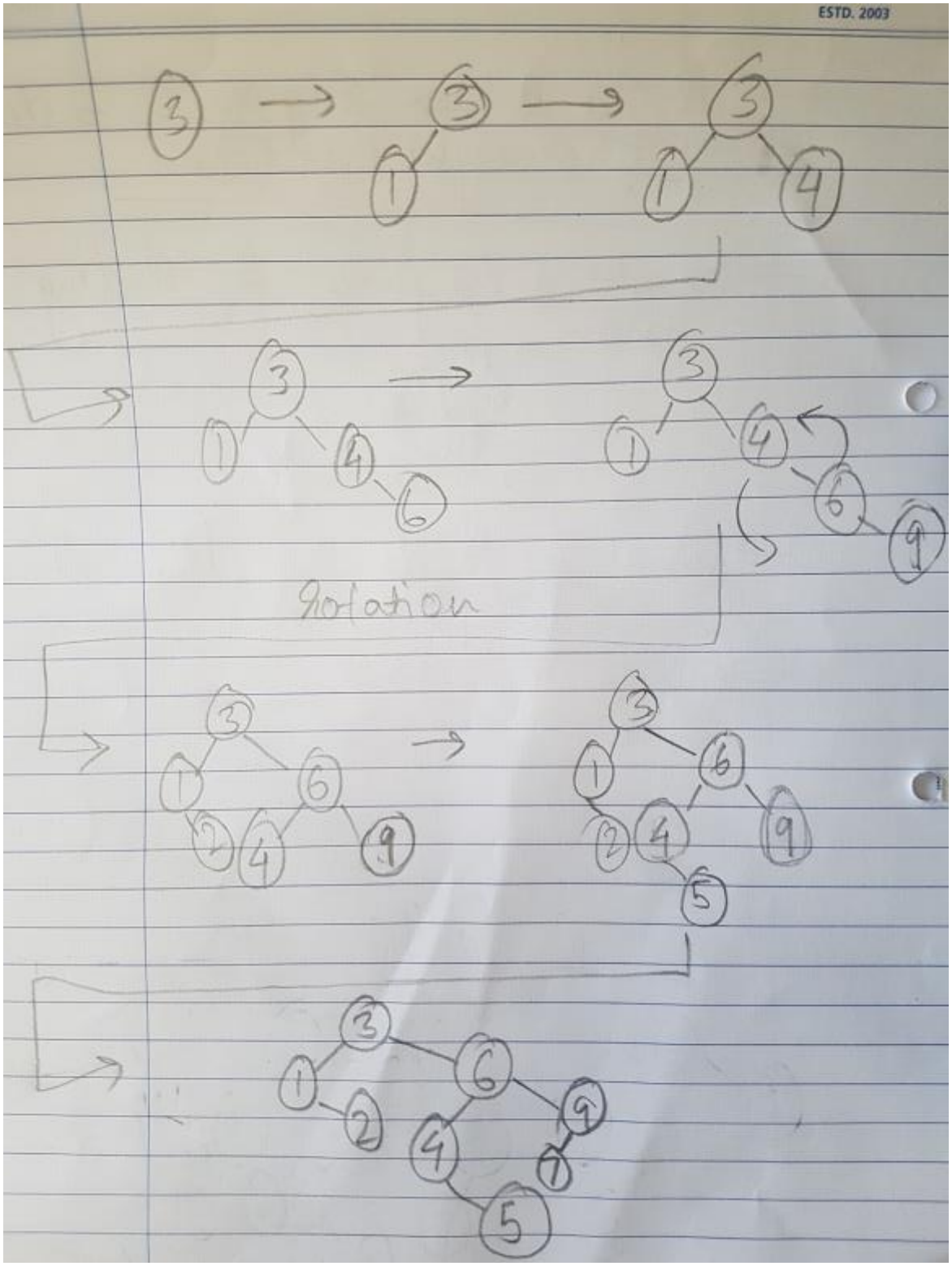
Q2b

Deletion



Q3

Avl insert



Q4

Preorder traversal gives us the rightmost node and the center node of t. Postorder traversal gives us the leftmost node and the center node.

However, even though we have the leftmost, rightmost, and center nodes, we cannot predict a binary tree:

Let the leftmost node be L, the rightmost node be R, the center node be C.
In the case where L is the only child of its parent, L can be the left child OR the right child. Similarly, if R is the only child of its parent, R can be the left child OR the right child. It is impossible for us to know the exact structure of t because we cannot know whether L is a left or right child, and whether R is a left of right child.

Q5

For lazy deletion, the BinaryNode class must now implement another instance variable of the Boolean type that will be true if the node has been deleted and false if not. The constructor of BinaryNode will set this variable to false when the node is instantiated.

```
public static int findMin(BinaryNode root) {
	if(root == null){
		return Integer.MAX_VALUE;
	}
	int left = findMin(root.left);
	if(!root.deleted) {
		if(left < root.data){
			return left;
		} else {
			return root.data;
		}
	}
	return findMin(root.right);
}
```