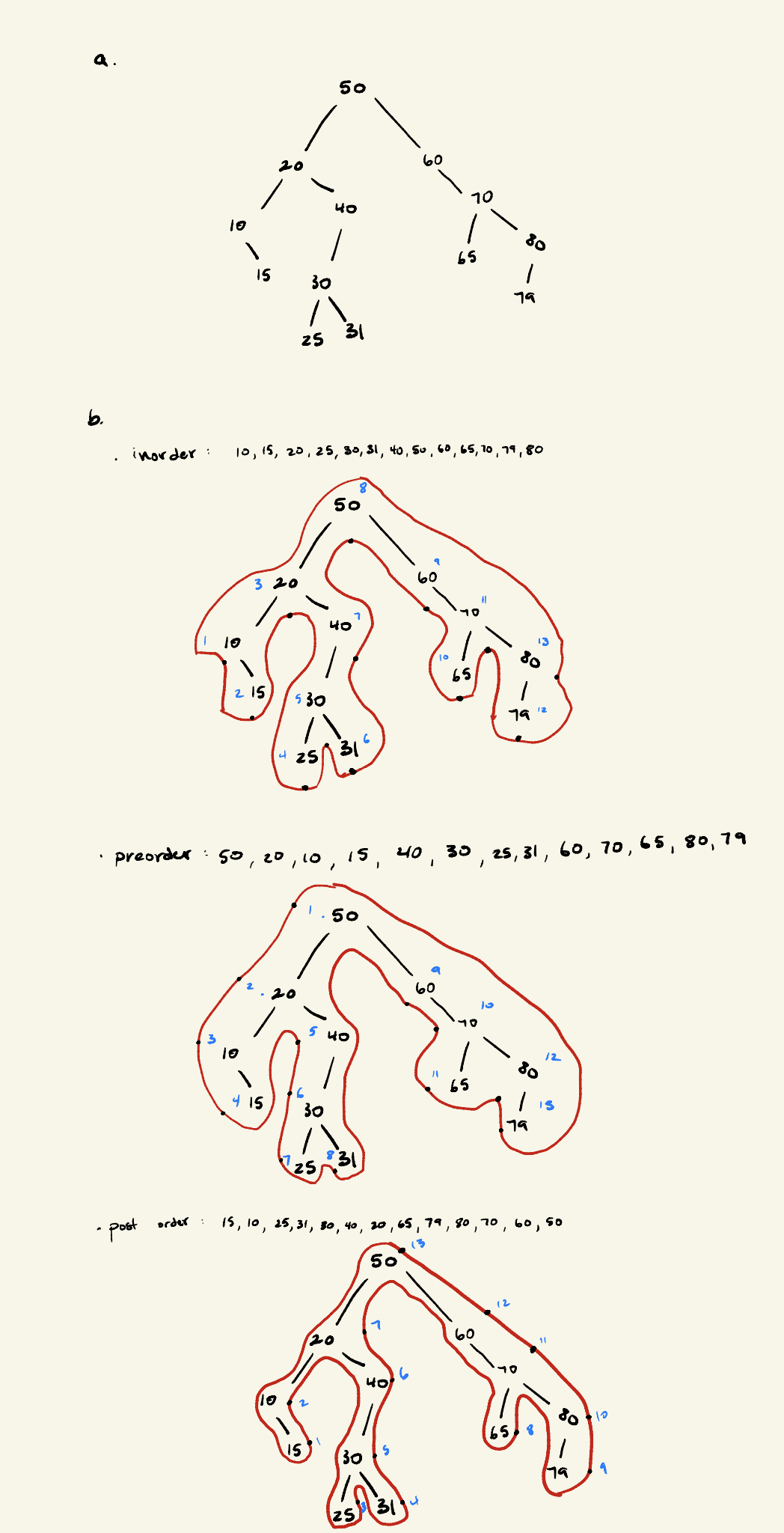
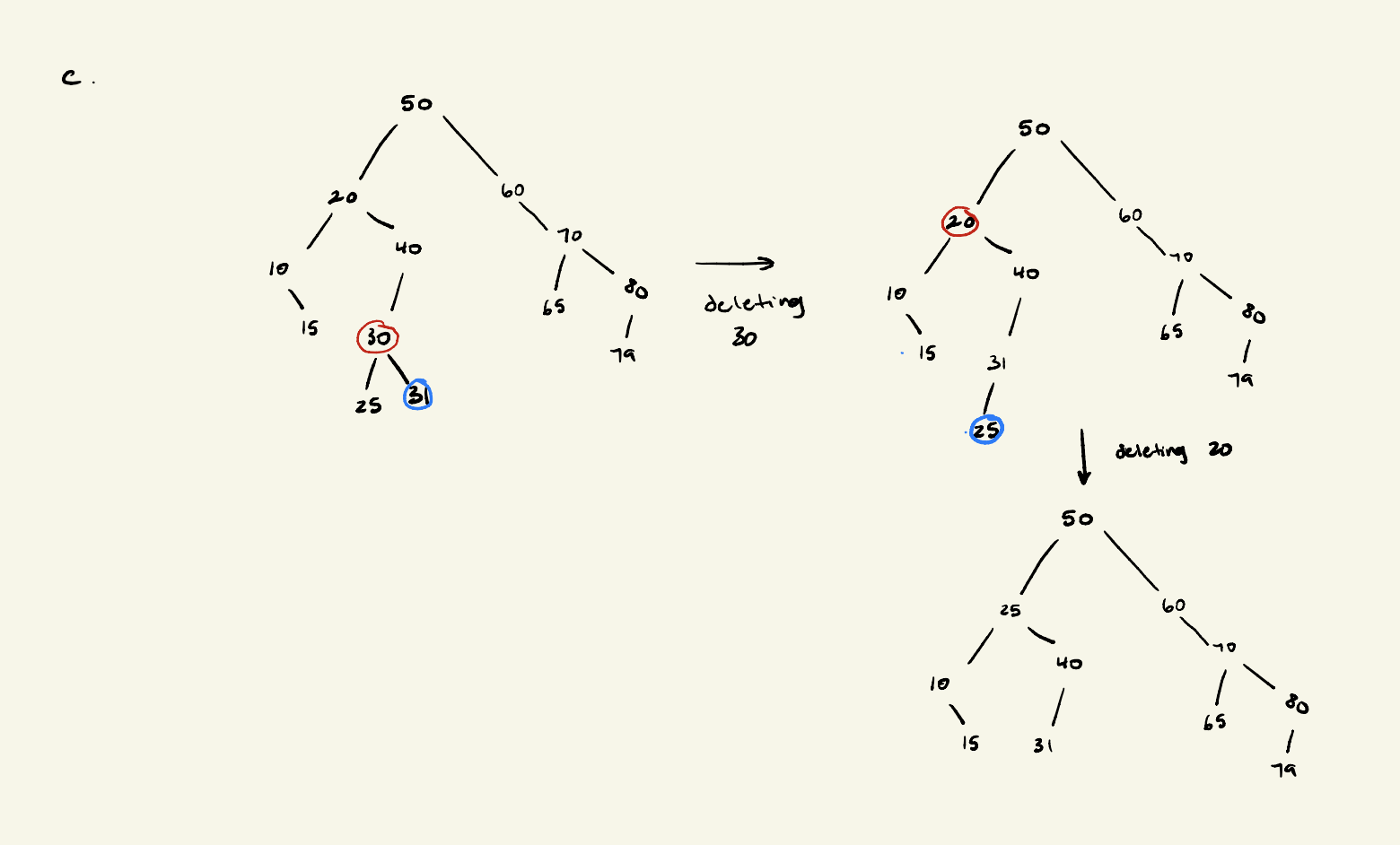
# Hw5 Yash Shah

### 1





### 2

#### 2a.

struct Node{

Node\* parent;

Node\* leftChild;

Node\* rightChild;

int val;

}

#### 2b.

If tree is empty / root is a nullptr

Create a node ptr parent and set it as the root with value set to inserted val and its left and right children set to nullptrs

Create node ptr current and set it to the root of the tree

Create node ptr parent and set it to a nullptr

While current ptr is not a nullptr

Set parent to current

If val > current’s val

current is set to current’s right child

else

current is set to current’s left child

create new node toInsert and set its value to val inserted

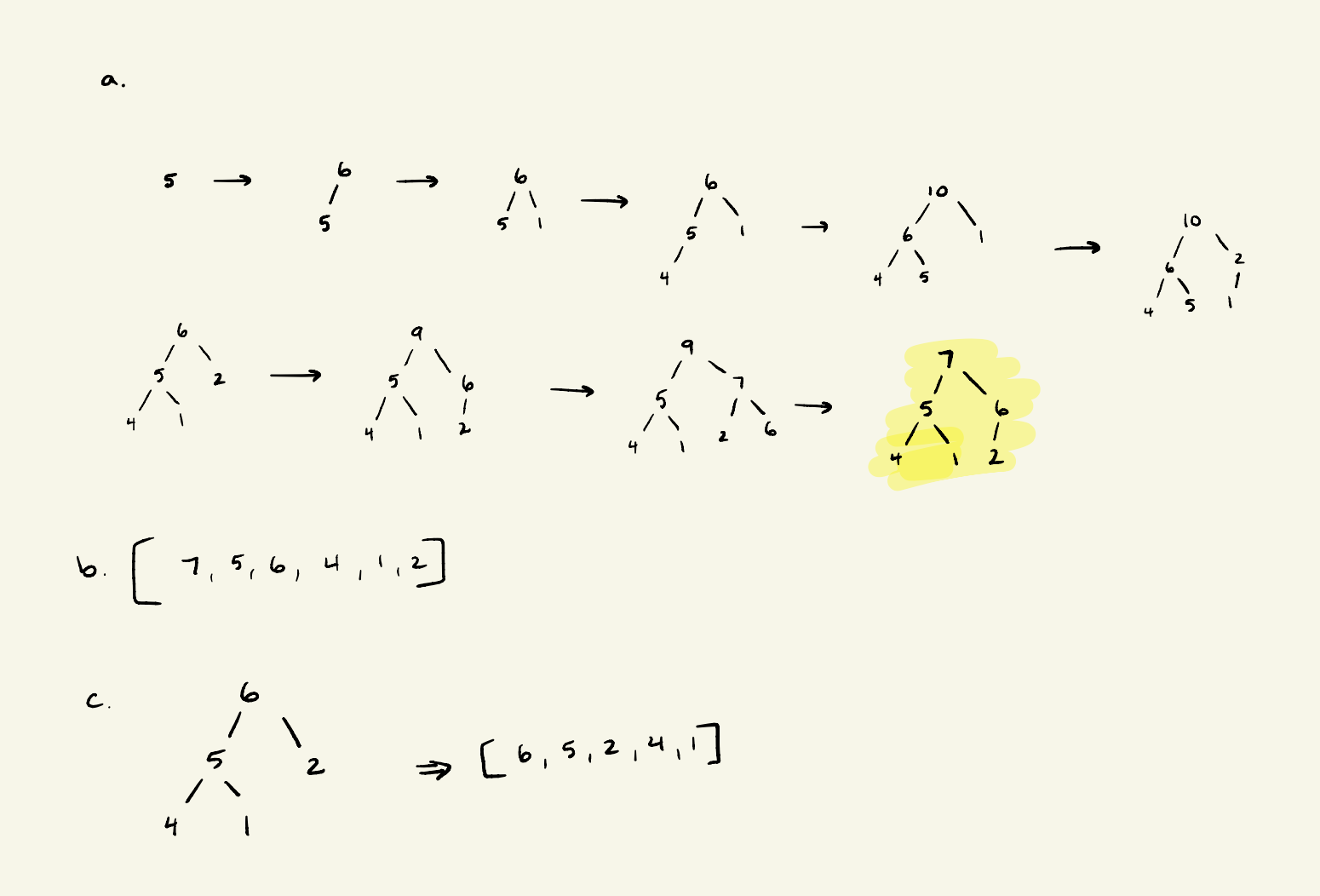
if val > parent’s val

set parent’s right node to toInsert

else

set parent’s left node to toInsert

### 3

****

### 4

#### 4a.

Linear search through the outer vector to find course c, linear search as well to find student s 🡪 O(C) + O(S) = **O(C+S)**

#### 4b.

Binary search tree to search through map to find course c, linear search to find student s 🡪 O(log(C)) + O(S) = **O(log(C) + S)**

#### 4c.

Binary search tree to search through map to find course c, binary search tree to search through set to find student s 🡪 O(log(C)) + O(log(S)) = **O(log(C) + log(S))**

#### 4d.

Searching through hash table to find course c, binary search true to search through set to find student s 🡪 O(1) + O(log(S)) = O(log(S) + 1) = **O(log(S))**

#### 4e.

Searching through hash table to find course c, searching through has table to find student s 🡪 O(1) + O(1) = O(2) = **O(1)**

#### 4f.

Searching through binary search tree to find course c, set is organized as a balanced binary search tree so writing each id numbers of all students in sorted order will be be done in time O(S) 🡪 O(log(C)) + O(S) = **O(log(C) + S)**

#### 4g.

Searching through a hash table for a course c, and an unordered set organized as a hash table that needs to be sorted (O(log(S))) and then printed (O(S)) 🡪 O(1) + O(S) \* O(log(S)) = O(1) + O(S\*log(S)) = **O(S\*log(S))**

#### 4h.

Since the hash table of unordered\_map helps us only to find the courses not the students, we have to parse through each course and check through the set of students and print for each course if a student s is taking that course by parsing through its binary tree 🡪 O(C) \* O(log(S)) = **O(C\*log(S))**