CS5343 Algorithms and Data Structures

Midterm Exam – Jun 26, 2019

PLEASE USE PEN and NOT PENCIL

First Name:_____ Last Name:_____

Student #:_____

Note: In all programs, do not use global variables, static variables or add new arguments to function calls.

Note: In all questions show steps.

- Write an Append() Algorithm that takes two single-linked lists, 'a' and 'b', appends 'b' onto the end of 'a'. You are given the head pointer of each list. Any of the two lists could be empty. (5 Points)

```
Node {
   Int key;
Node *next;
}

Main() {
// init heada and headb as nodes)
        Append(heada, headb);
}

// answer
Append(heada, headb) {
        Node current = heada;
        while(current.next != null) {
                current = current.next;
        }
        current.next = headb;
}
```

- Given the Node structure:

```
public class IntNode
{
        int data;
        IntNode *next;
        IntNode *prev;
}
```

You are given a pointer to the head of the list, and a pointer to the tail of the list. Assume the list is not empty has greater than 2 nodes and the number of nodes is Odd.

Write a recursive algorithm that would delete the middle node (remember the nodes are odd, so there is a middle node). You cannot traverse the list to count the number of nodes. (10 points)

```
Main() {
    Delete( head, tail):
}

// answer
void Delete(head, tail) {
        while(head != tail) {
                head = head.next;
                tail = tail.prev
        }
        // now head == tail, both pointing to middle node
        IntNode prev = head.prev;
        IntNode next = head.next;
        prev.next = next;
        next.prev = prev;
}
```

- Recursive algorithm:
    - Write a recursive method named count_nodes that returns the number of nodes in a Binary Tree. (8 points)
    - What is the Big O of your algorithm?(2 points)

//answer

```
Count_nodes( Node *node) {
if(node == null) return 0;
else return 1 + count_nodes(node.left) + count_nodes(node.right);
}
```

BIGO: O(n) - have to visit every node

- Write an algorithm to do inorder traversal in a binary tree. (5 points)

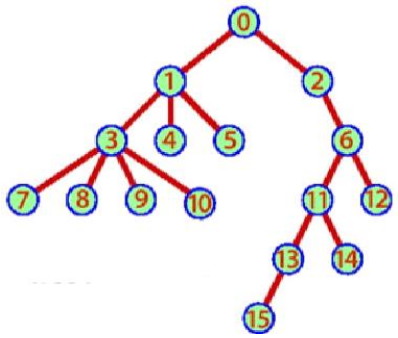
```
void inOrderTraversal(Node node) {
  if(node == null) return;
inOrderTraversal(node.left);
System.out.print(node.val + " ");
inOrderTraversal(node.right);
}
```

- Consider the following pseudo code:

```
declare a stack of characters
while ( there are more characters in the word to read )
{
        read a character
        push the character on the stack
}
while ( the stack is not empty )
{
        pop a character off the stack
        write the popped character to the screen
}
```

  What is written to the screen for the input "imagined"? (3 points)
  Assuming characters are read in from left to right: denigami

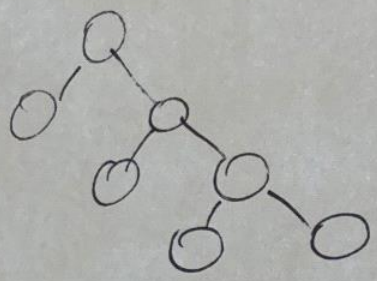
- Given the following tree: (5 points)
  - Which node has the largest depth? 15
  - Which node or nodes has/have the smallest depth? 0 (root)
  - What is the height of the tree? 5
  - Which nodes have a height of 2? 7,8,9,10,11,12
  - Which nodes have a depth of 2? 3,4,5,6



- Show an example of full binary tree that is not complete.(2 points)

  << picture >>

Full but not complete



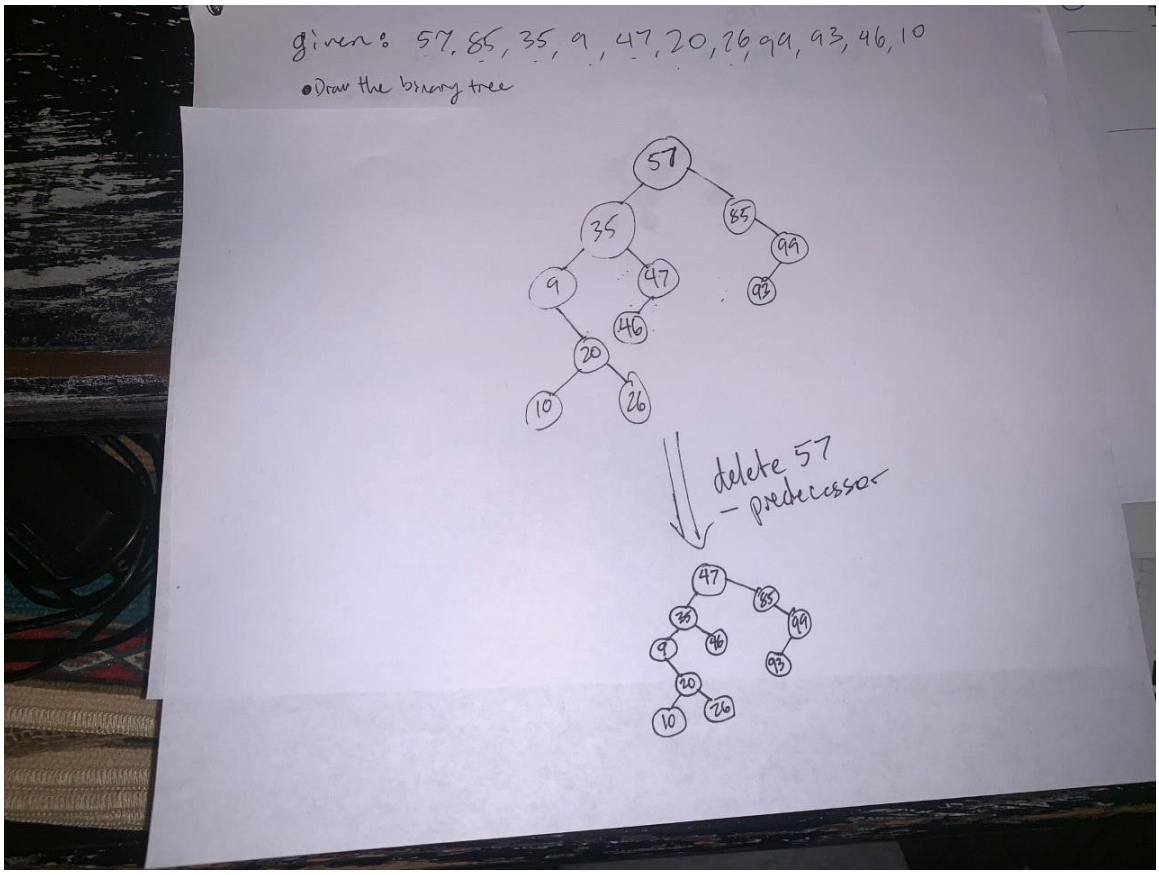
- Given a complete binary tree of height h, (think carefully!!!)

  What is the maximum number of leaf nodes? (3 points)

  2^h-1

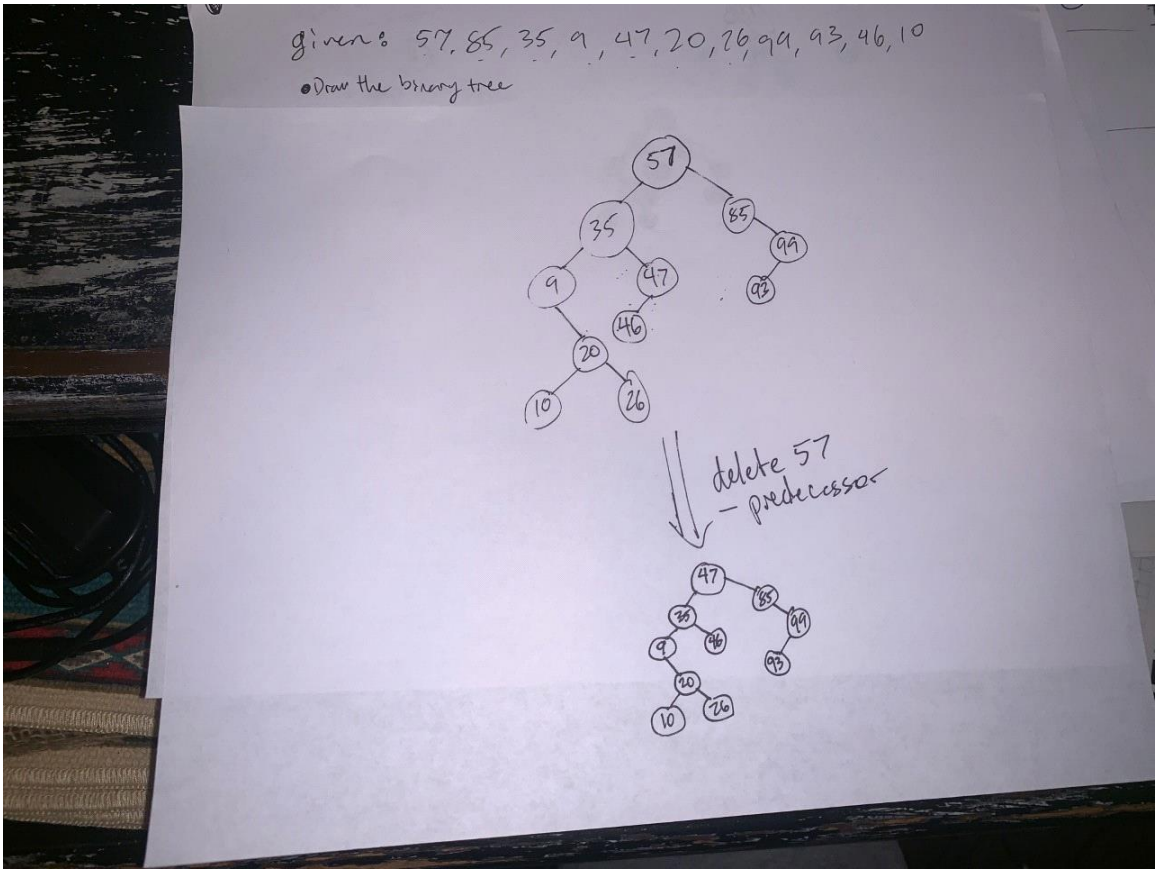
- Given the following integers: 57, 85, 35, 9, 47, 20, 26, 99, 93, 46, 10
  - Draw the binary search tree that results from inserting the integers in the given order.(8 points)

    << PICTURE>>

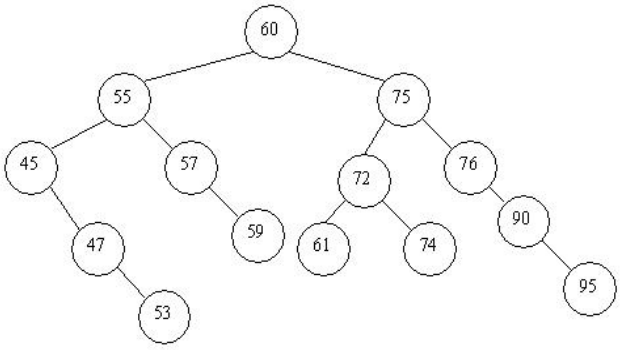


- Draw the resultant tree after deleting 57 (3 points).

<< picture>>



- Given the following tree:



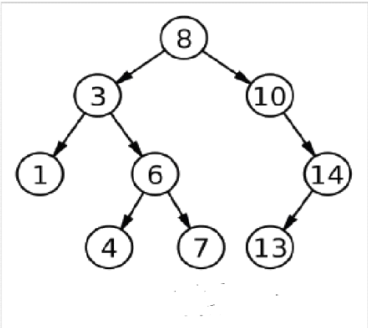
  - Show the post-order traversal.(2 points)
  - 53,47,45,59,57,55,61,74,72,95,90,76,75,60

  - Show the pre-order traversal.(2 points)
  - 60,55,45,47,53,57,59,75,72,61,74,76,90,95

  - Show the in-order traversal.(2 points)
  - 45,47,53,55,57,59,60,61,72,74,75,76,90,95

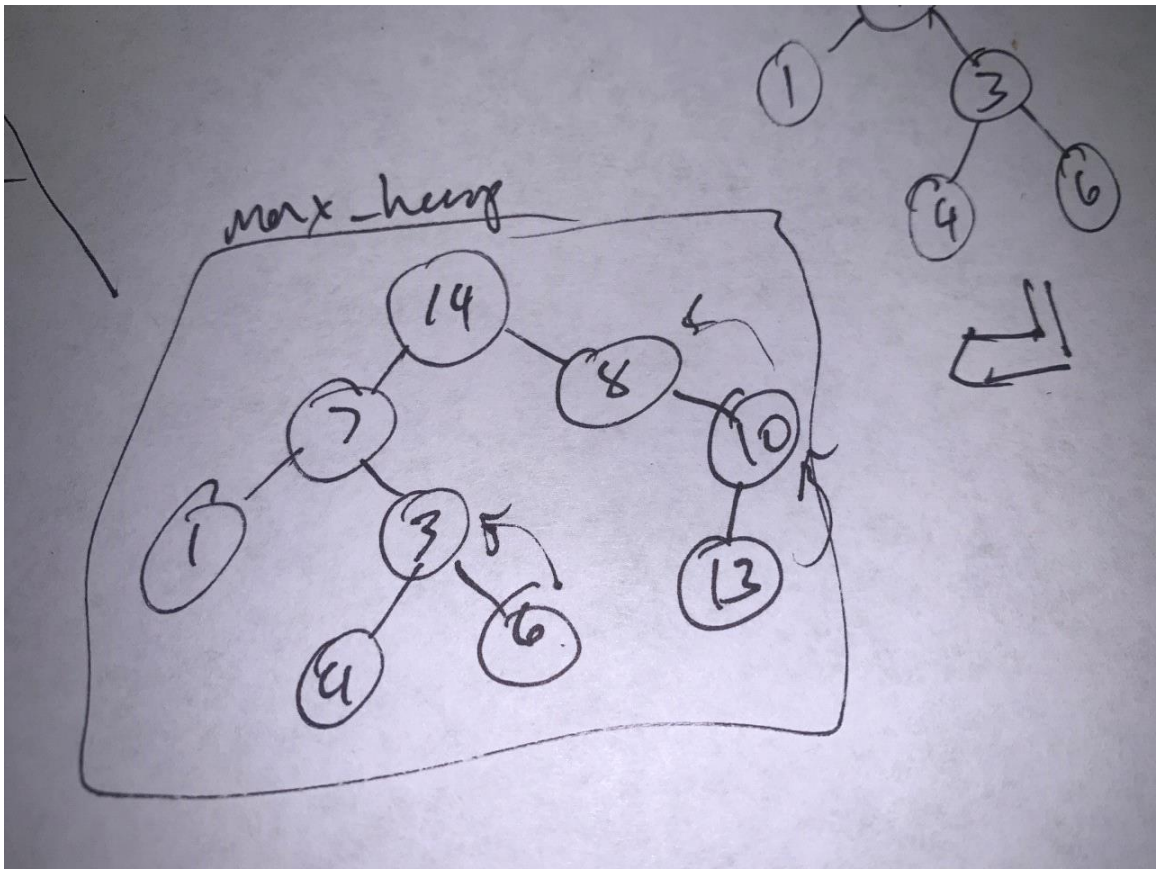
  - Show the BFS traversal.(2 points)
  - 60, 55,75,45,57,72,76,47,59,61,74,90,53,95

- Given the tree below:



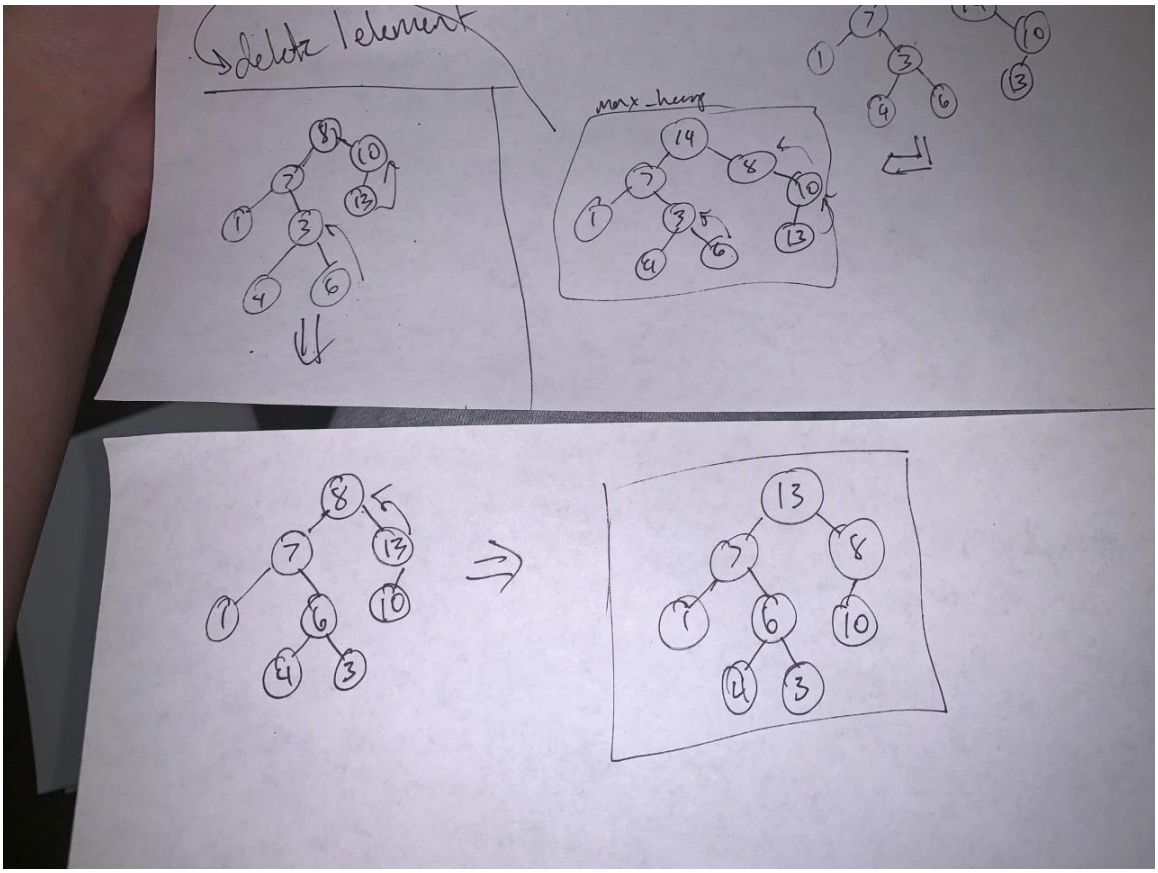
  - Convert it into max heap (8 points).

    << picture >>



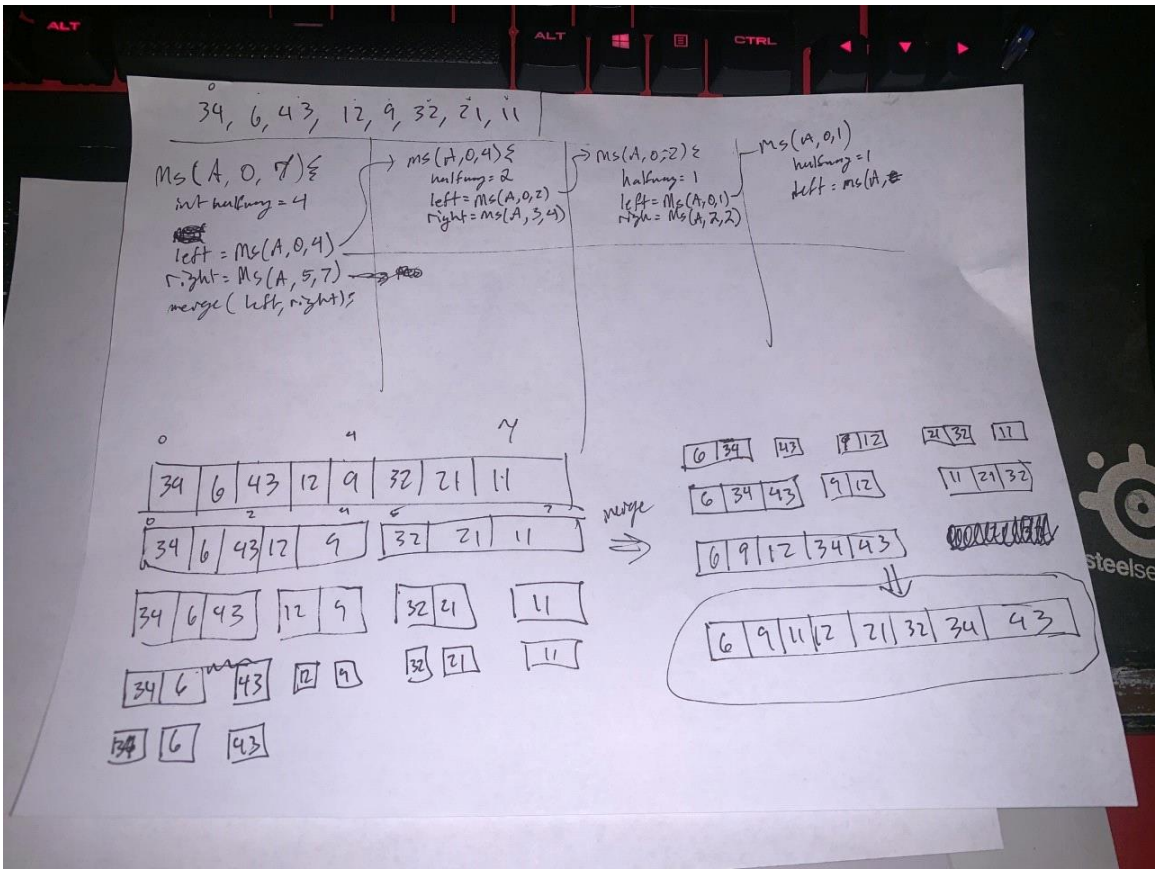
- Show the heap after you delete one element (array is fine or you can draw the tree) (2 points)

  << picture >>



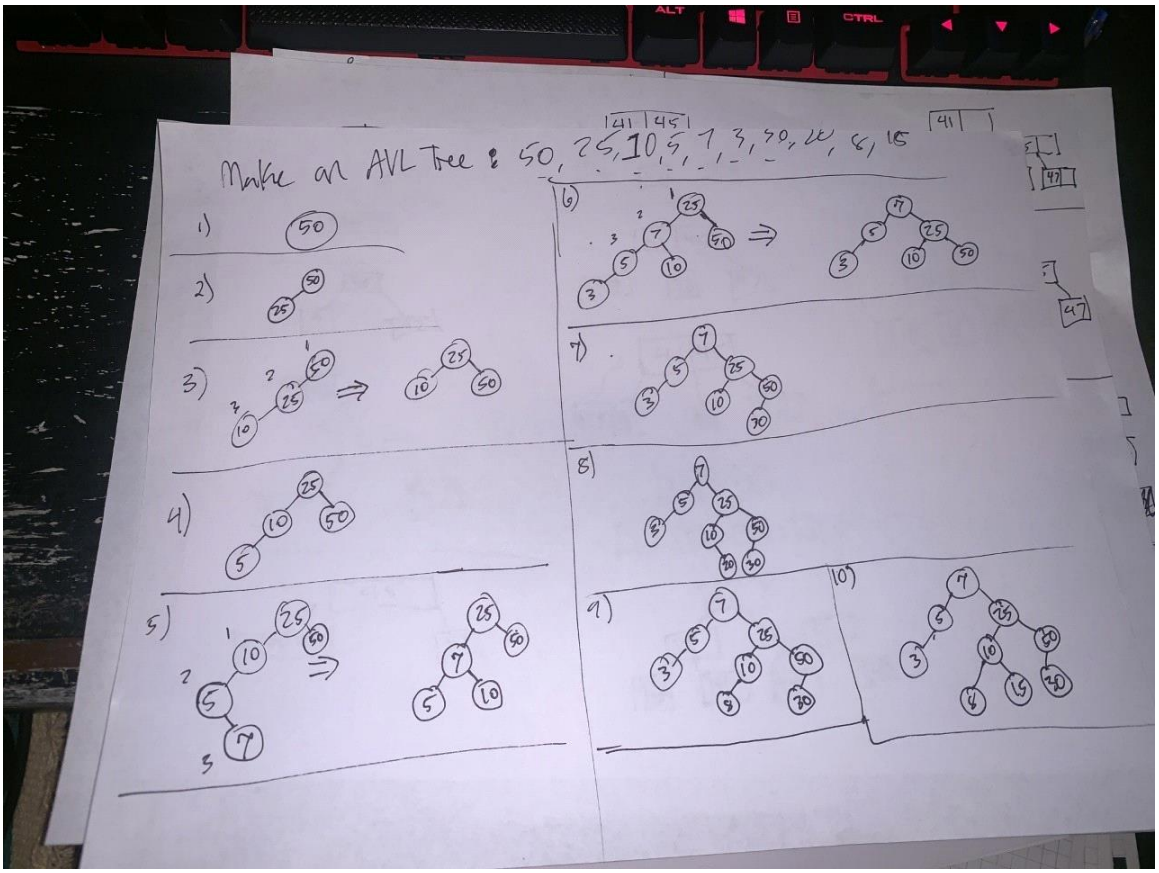
- Given the following integers: 34, 6, 43, 12, 9, 32, 21, 11;
    - Sort them in ascending order using merge Sort method (8 points)

<< picture >>



- Given the following integers: 50, 25, 10, 5, 7, 3, 30, 20, 8, 15
  Make an AVL tree as you insert each number in order.(10 points)

  << picture>>


Make an AVL Tree : 50, 25, 10, 5, 7, 3, 30, 40, 8, 15
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)

- Given the following keys
    - 9 47 45 43 41 39 37 35 33 31

Draw the Btree of order 3 for the following numbers, inserted in order. (10 points)

<< picture>>