Key

CS 435 – Advanced Data Structures

Test #2

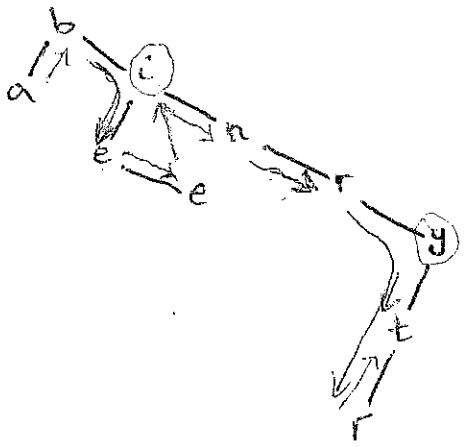
1. Insert the following keys in a binary tree: **b i n a r y t r e e**, using:

   a. In-order traversal:

   L → N → R

   10 keys

   a b e e i n r r t y

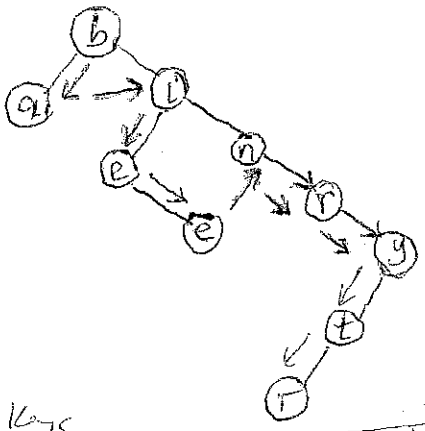


   b. Pre-order traversal:

   N → L → R

   10 keys

   b a i e e n r y t r

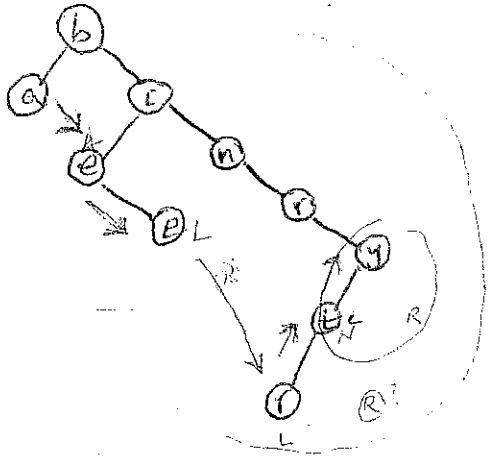


   c. Post-order traversal:

   L → R → N

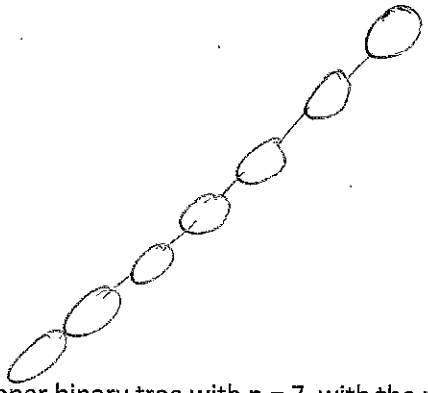
   10 keys

   a e e r t y r n i b



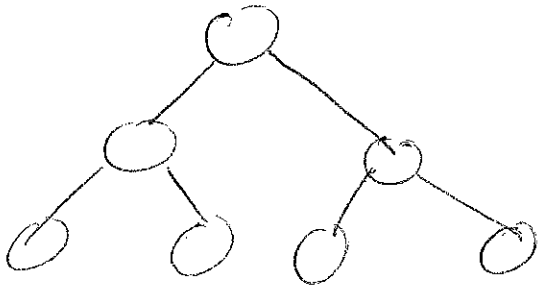
2. What are the minimum and maximum number of internal and external nodes in an improper binary tree of n nodes? Answer 'a' and 'b' with n = 7.

a. Show an improper binary tree with n = 7, with the maximum number of internal nodes.



Internal nodes = 6 (max)
External nodes = 1
total = 7

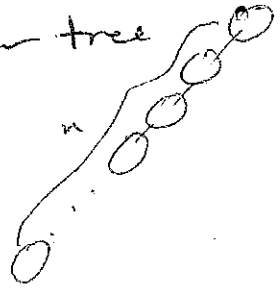
b. Show a proper binary tree with n = 7, with the minimum number of internal nodes.



Internal nodes = 3 (min)
External nodes = 4
total = 7

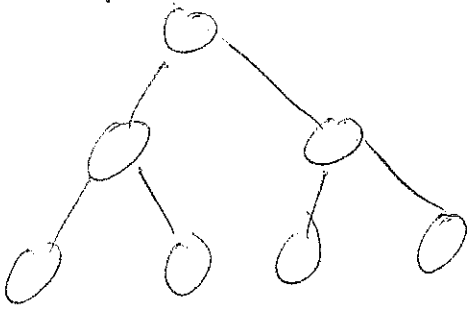
3. What are the minimum and maximum number of internal and external nodes in an improper binary tree of n nodes? Your answer must be for a binary tree of n nodes.

a. Maximum Improper tree



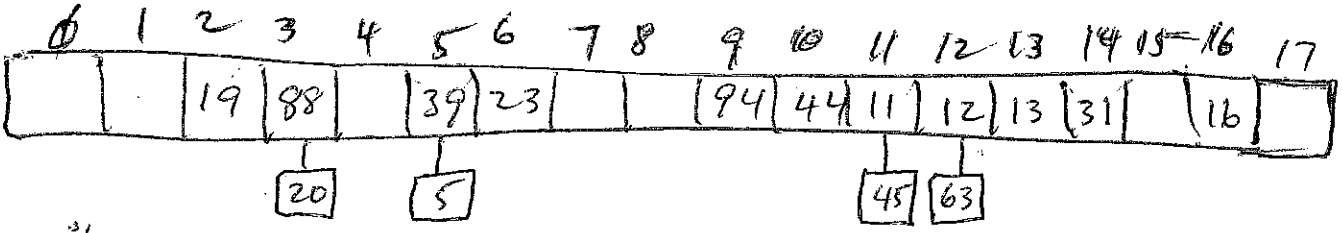
Internal nodes = n − 1
Ext. nodes = 1
total = n

b. Minimum Proper tree



Internal nodes = $\frac{n-1}{2}$
Ext. nodes = $\frac{n+1}{2}$
total = n

4. Draw the 18-entry hash table that results from using the hash function: $h(k) = k \bmod 17$, to hash the keys 12, 44, 13, 88, 23, 94, 11, 45,19, 63, 31, 39, 20, 16, and 5, assuming collisions are handled by chaining.



12 % 17 = 12

44 % 17 = 10

13 % 17 = 13

88 % 17 = 3

23 % 17 = 6

94 % 17 = 9

11 % 17 = 11

45 % 17 = 11

19 % 17 = 2

63 % 17 = 12

31 % 17 = 14

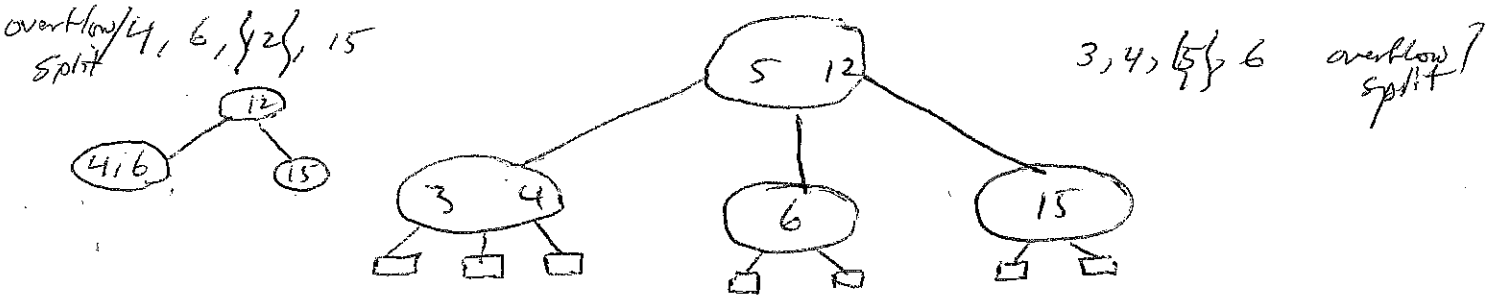
39 % 17 = 5

20 % 17 = 3

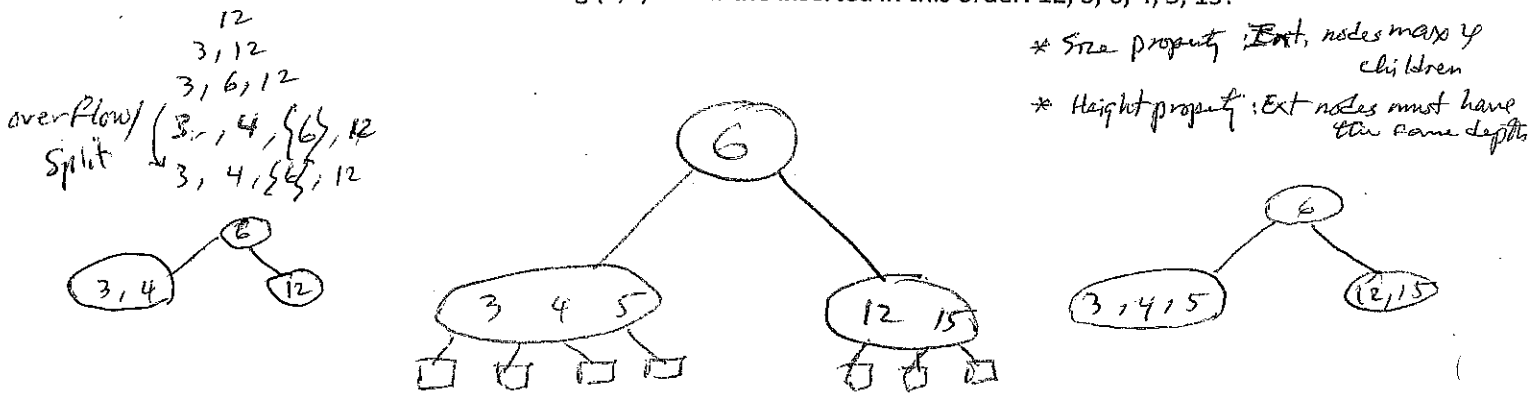
16 % 17 = 16

5 % 17 = 5

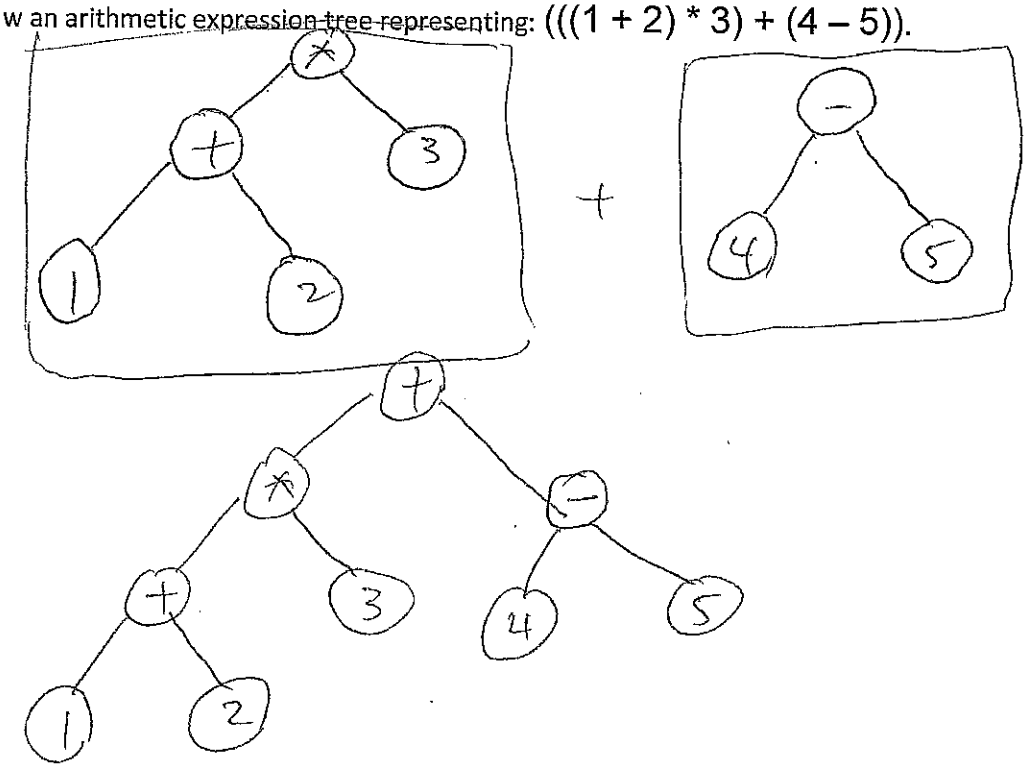
5. A. Store in a (2, 4) tree the following (in this order): 4, 6, 12, 15, 3, 5 and show the resulting tree.



B. What would be the resulting (2,4) tree if the inserted in this order: 12, 3, 6, 4, 5, 15?

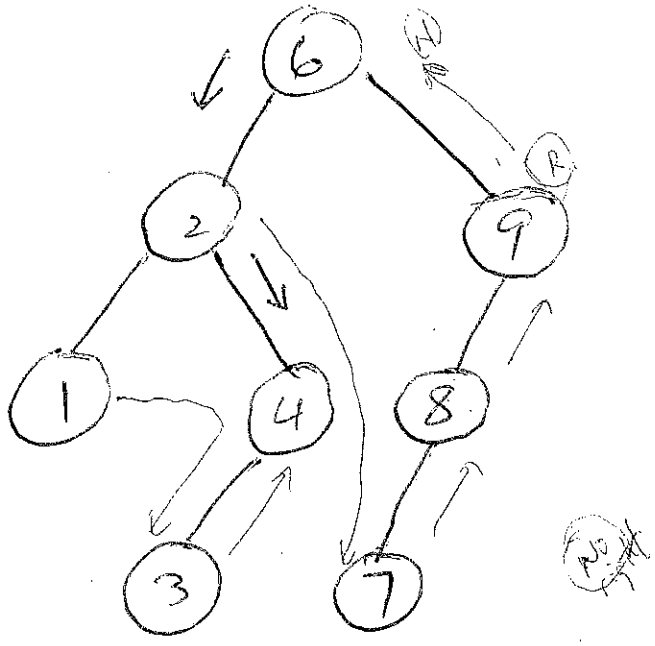


6. Draw an arithmetic ~~expression tree representing~~: (((1 + 2) * 3) + (4 – 5)).



7. The values 6 2 1 9 8 4 3 7 are inserted into a binary tree in the given order.
   a. Draw the resulting binary search tree.
   b. Explain the comparisons that would be needed to determine 5 was not in the tree.
   c. Give the order in which the nodes are visited in inorder.
   d. Give the order in which nodes are visited in postorder.

a.



b. 5 < 6 , check left subtree

5 > 2 , check right subtree

5 > 4 , check right subtree

4 does not have a right subtree
(5 does not exist)

c. 1 2 3 4 6 7 8 9

d. 1 3 4 2 7 8 9 6

↳ → R → N

8. Review the following version of findIndex method of the SortedTableMap class,

```
1   private int findIndex(K key, int low, int high) {
2     if (high < low) return high + 1;
3     int mid = (low + high) / 2;
4     if (compare(key, table.get(mid)) < 0)
5       return findIndex(key, low, mid − 1);
6     else
7       return findIndex(key, mid + 1, high);
8   }
```

a. Does this always produce the same result as the original version? For example, a call of **findIndex(20, 0, 2)** for a table with contents **{10, 20, 30}** would return 1 as answer. What would be the return value for this version?

initial call ← rf(20, 0, 2) → m = (0+2)/2 = 1 ✓
(4) (20 − 20) < 0 ? NO.
rf(20, 2, 2)
m = (2+2)/2 = 2 ✓
(4) (20 − 30) < 0 ? yes.
↳ rf(20, 2, 1) ⇒ (2) if (1 < 2) return 1+1 = [2] ← i.e., (30)

b. How would you modify the above version so that it would generate the correct result?

Make (4) if (compare(key, table.get(mid)) <= 0)
↑ modification

m = 1
(4) (20 − 20) == 0 ? True
rf(20, 2, 0)
low high
(2) (0 < 2) ? yes
⇒ return 0+1 = [1] ← i.e., (20) ✓