**WEEK - 5**

**R-8.1**

* . /user/rt/courses/ → root.
* /user/rt/courses/,cs016/, cs252/, homeworks/, programs/, projects/, papers/, demos/ → Internal Nodes
* &ndash; grades, homeworks/, programs/, hw1, hw2,hw3, pr1, pr2, pr3 → 9 descendants.
* /user/rt/courses/ → 1ancestor
* grades, programs/
* papers/, demos/, buylow, sellhigh, market
* Thedepth is 3, because it has 3 ancestors

**R-8.4**

* children(p) will iterate through the position p O(Cp + 1)time. So the running time of a call to T.height depends upon the O(Cp+1)time.

**R-8.20**

Might Try:

E

/ \

X F

/ \ / \

A M U N

Solves Pre-order, but fails on in-order

Preorder of EXAMFUN mandates E be the root

Inorder of MAFXUEN mandates M be the leftmost leaf

Next try:

E

/

X

/ \

A F

/ / \

M U N

Fails again on in-order, try again

E

/ \

X N

/ \

A U

/ \

M F

And there you have it.

**C-8.42**

1) Recursively traverse the given Binary and copy the key-value, left pointer, and a right pointer to the clone tree. While copying, store the mapping from the given tree node to clone the tree node in a hashtable. In the following pseudo-code, ‘cloneNode’ is the currently visited node of the clone tree and ‘tree node’ is the currently visited node of the given tree.

2) Recursively traverse both trees and set random pointers using entries from the hash table.

**R-9.3**

(1,D), (3,J), (4,B), (5,A), (2,H), (6,L).

**R-9.5**

If n is odd then initialize min and max as the first element.

If n is even then initialize min and max as minimum and maximum of the first two elements respectively.

For the rest of the elements, pick them in pairs and compare their

maximum and minimum with max and min respectively.

**R-9.21**

Replace the root or element to be deleted with the last element.

Delete the last element from the Heap.

Since the last element is now placed at the position of the root node. So, it may not follow the heap property. Therefore, heapify the last node placed at the position of the root.

**C-9.26**

// stack of Key

class Stack {

class Element { int prio, Key elem; };

MaxPriorityQueue<Element> q;

int top\_priority = 0;

void push(Key x) { q.push(Element(top\_priority++, x)); }

Key pop() { top\_priority--; return q.pop().elem; }

};