Omar Khattab

Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Assignment 3 (3% - 12 points)

CSI2110/CSI2510 (Fall 2021)

Due: Thursday Oct 7, 11:59PM

Late assignment policy : ***1min-24hs late are accepted with 30%off; no assignments accepted after 24hs late.***

Question 1. **[4 points = 2+2] Give the answers and justify.**

a) Give the last 4 digits your student number *sn*=..... How many **internal nodes** and **external nodes** are there in a **full binary search** tree with n nodes, where n = *sn, if sn is odd, and* n=*sn+1, if sn is even* ? 2727 n=2727 (number of nodes)

n = 2e-1 (where e is no. of leaves)
e = (n+1)/2 = (2727+1)/2 = 1364 leaves

e = i+1 (where i is no. of internal nodes)
i = e-1 = 1364-1
= 1363 internal nodes

2727 - 1363 = 1364 external

b) Let $k \geq 2$. What is the height (as a function of k) of a **complete binary tree** with 3×2^k nodes ?

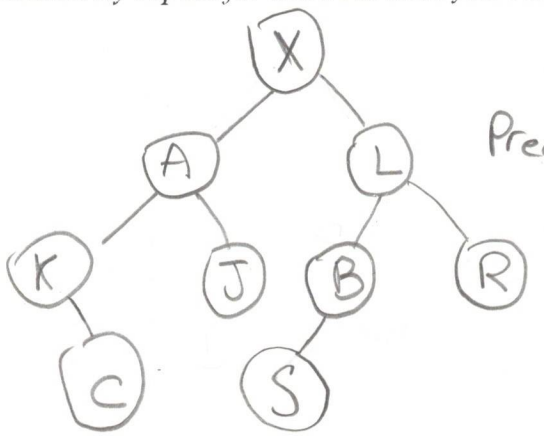
$h = \lfloor \log_2 3 \times 2^k \rfloor$ ← floor

Question 2. **[2 points]**

For a binary tree, its inorder traversal gives (K, C, A, J, X, S, B, L, R) and its postorder traversal gives (C, K, J, A, S, B, R, L, X) .

Draw the tree and give its preorder traversal.

Hint: Alternatively examine the postorder and inorder to go discovering root, elements on left and elements on right, and recursively repeat for subtrees until you build the whole tree.



Preorder (X, A, K, C, J, L, B, S, R)

Question 2

```
TreeNode DeepstAncestor (TreeNode p, TreeNode q){
    TreeNode ancestor; Boolean found=false;
    if (P.parat == q.parat)
        return P.parat

    int pDepth=0, qDepth=0;
while (P.parent != null){
    pDepth++;
    P= P.parent;
}
while (q.parent != null {
    qDepth++;
    q = q.parat
}
if (qDepth == pDepth){
    for (int i=0, i ≤ qDepth, i++){
        if (P.parat == q.parat){
            ancestor = p.parat;
        }else{
        P=P.parat;
        q=q.parat;
    }
}else if (qDepth > pDepth){
    for (int i=0, i ≤ qDepth, i++){
        if (q.parat == P){
            ancestor = P; found = true
        }else{
        q=q.parent;
    }
if (found == false){
    while (qDepth != pDepth){
        q=q.parat; qDepth--;
    }
(( same code goes here ))
}
```

continue here

```
else if (pDepth > qDepth
    for (int i=0, i ≤ pDepth, i++
    if (P.parat == q){
        ancestor = q; found=t
    }else{
    P=P.parat;
    }
if (found == false){
    while (pDepth != qDepth
        P=P.parat; pDepth--
    }
(( same code goes here
}
return ancestor;
}
```