```cpp
/**
 *  OPPGAVE A:
 *     Skriver annenhver bladnode startende fra høyre, og med den andre noden.
 *     Traverserer: "seg selv", "høyre node", "venstre node".
 *   @param   node  -  Noden som skal besøkes.
 */
void skrivAnnenhverBladnode(Node *node)
{
    if (node) // sjekker om noden peker til en reel node.
    {
        if (!node->left && !node->right && !skrev) // sjekker om noden ikke har noen barn dvs "er en blad node".
                                                   // og om den tidligere noden har allerede skrevet ut. skrev variabel starter som true.
        {
            cout << node->ID << " ";
            skrev = true; // setter skrev til å være true etter å hatt skrevet.
        }
        else if (!node->left && !node->right) // hvis noden er en blad node, men tidligere bladnode har skrevet. skrev blir da satt til false.
        {
            skrev = false;
        }
        else
        { // hvis noden ikke er en blad node fortsetter koden med høyrebarn først.
            skrivAnnenhverBladnode(node->right);
            skrivAnnenhverBladnode(node->left);
        }
    }
}

/**
 *  OPPGAVE B: Finner antall noder under 'node' STØRRE ENN enn 'verdi'.
 *  Traverseres på en preorder måte ikke alle nodene blir besøkt.
 *  @param   node   -  Noden som skal besøkes/undersøkes
 *  @param   verdi  -  verdien det skal sjekkes om nodens ID er større enn
 *  @return  Antall noder under 'node' med 'ID' STØRRE ENN 'verdi'
 */
int tellStorre(Node *node, int verdi)
{
    int antall = 0; // antall noder større enn verdi.
    if (node && node->ID > verdi) // sjekk hvis node eksiterer og om den er større en vedien.
    {
        antall++; // inkrementerer antall med 1 hvis noden er større enn verdi.
        antall += tellStorre(node->left, verdi); // inkrementerer antall som blir funnet fra venstre og høyre node.
        antall += tellStorre(node->right, verdi);
    }

    return antall; // returnerer totalt antall noder større enn verdi.
}
```

```cpp


/**
 *  OPPGAVE C: Finner ut om alt under 'node' er mindre dens 'ID'.
 *  Går gjennom på en postorder måte om en av barna er større returnere funksjonen.
 *  @param   Node  -  Noden som skal besøkes/undersøkes
 *  @return  Returnerer om 'node's ID er større enn barnas eller ei
 */
bool storreEnnBarna(Node *node)
{
    bool left = true;
    bool right = true;
    if (node) // sjekker om det er en node
    {
        if (node->left) // sjekker om venstre barn eksisterer
        {
            if (node->left->ID >= node->ID) // hvis venstre barn har høyere eller lik verdi til nåverende node returneres false.
            {
                return false;
            }

            else // ellers fortsetter koden og sjekke storreEnnBarna for venstre barnet.
            {
                left = storreEnnBarna(node->left);
            }
        }

        if (node->right) // sjekker om venstre barn eksisterer
        {
            if (node->right->ID >= node->ID) // hvis høyre barn har høyere eller lik verdi til nåverende node returneres false.
            {
                return false;
            }
            else
            {
                right = storreEnnBarna(node->right); // ellers fortsetter koden og sjekke storreEnnBarna for høyre barnet.
            }
        }
    }
    return left && right; // returnere om venstre side og høyre side er har barn større enn foreldre.
}
```