**Breadth-Last Traversal of a Binary Tree – 40 points;**
**Suggested work & submission time less than 40min**

Submission instructions:
Please submit a single C file (use any file name) on TEACH under FinalExamProblem1 before 10:30am.
We will not accept hand-written solutions.

Implement a C function, `printBreadthLastBinaryTree()`, that prints out node values of a given binary tree in the breadth-last order, such that every tree level is printed in a new row, bottom-up, starting from the leaf level to the root, as illustrated in Figure 1.

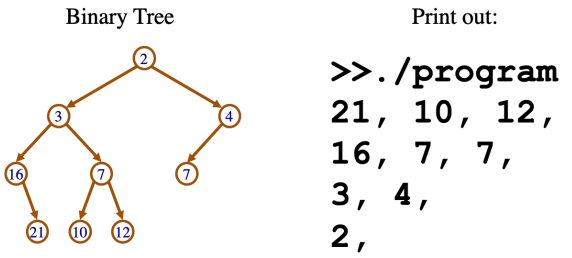


Figure 1: Breadth-last traversal of a binary tree. The function prints out node values in separate rows for every tree level, bottom-up, starting from the leaf level to the root.

The input argument is a pointer to a binary tree. The function has a **strict memory constraint** that absolutely no new data structures can be formed and no new memory locations can be allocated, except for a couple of local integer variables. Also, time complexity of the function should not exceed $O(n)$, where $n$ is the number of nodes in the tree. The input binary tree and nodes in the tree are defined using the following `struct` data types:

```
#define TYPE double
struct Node {
  TYPE   val;         /*node's value*/
  struct Node *left;  /*pointer to the left child*/
  struct Node *right; /*pointer to the right child*/
};
struct Tree {
   struct Node *root;
   int size;
};
/* Breadth-last traversal of a binary tree
    - Input: tree = pointer to a binary tree.
    - Pre-conditions: tree was initialized well and exists in memory, but may be empty.
    - Constraints: time complexity <= O(n); no new data structures allowed,
                   no new memory locations can be allocated, except for a couple of integers
*/
void printBreadthLastBinaryTree(struct Tree *tree){
   assert(tree);

   /* FIX ME */
}
```

You should not modify the above definitions of `struct` data types, and you are allowed to add your own functions as you find appropriate. If your solution calls other functions, you will have to implement them for this problem. There is no need to implement the main function for this code.