Week 3: Heap & Priority Queue

We will keep developing your tree and tree node class in this week's lab, make sure you have some version control of your code.

Task 1: From linked list trees to array trees

Inside the tree class, create a function which can transform the linked list-based tree to an array.

To make it simple, you can assume the input tree is a complete binary tree. The tree store String objects, using them as the IDs and data. Suggested format of this method is

```
public String[] Tree2Array(Tree root)
```

Since you need to use breadth-first traversal algorithm, you may also include a queue. You can use previous developed doubly linked-list queue for this task.

Bonus: Create a function to check if input tree is a complete tree or not.

Task 2: Implement Priority Queue using Max-Heap

Create a separate Maximum Heap class using array structure (using lexicographical order). As we are using an array to store our data structure within the heap, you should grow it like a vector, as required, so the capacity is not limited. You should consider growing it level-by-level such that it grows just enough to hold the next complete level in the tree.

You should have the following methods in the heap class:

```
public boolean IsEmpty()          //return TRUE if the queue is empty

public int GetNoOfItems()         //return the number of items in the
                                  //queue

public void Push(String value)    //add item to the priority queue

public String Pop()               //return and removes the items at
                                  //the top of the heap
```