Homework 3: Playing with BinarySearchTree

You have been provided with an implementation of a $Binary\ Search\ Tree$. The following functions are already implemented in the file:

- $inserItem$: Inserts a new item in the binary search tree.
- $searchItem$: Searches for an item in the tree.
- $PrintInOrder$: Prints the tree by an in-order traversal of the tree.
- $calcHeight / calcNodeHeight$: Calculates the height of an item/node.

For this homework, you are required to add the following functions to the above implementation:

Task 1: Add getSize function. Add a new function $getSize$. This function returns then size of the tree. The size of a tree is the number of nodes in the tree. **You should write a recursive function for this task.**

Task 2: Add calcDepth/calcNodeDepth function. Add two new functions that will calculate the depth of a node and depth of an item. The function $calcDepth$ receives an item value and returns the depth of the item in the tree. The second function $calcNodeDepth$ receives a tree node and returns the depth of that node in the tree. See the $calcHeight$ and $calcNodeHeight$ functions for hints. **However, you cannot write recursive functions for this task.**

Task 3: Add getMinItem/getMaxItem function. Add two functions that will find and return the minimum and maximum item of the tree. **You cannot write recursive functions for this task.**

Task 4: Add rangeSearch function. Add a new function $rangeSearch$. This function receives three parameters: $node$, $leftBound$, and $rightBound$. The last two parameters indicate two integer values. The function should return the number of items in the tree that are greater than or equal to $leftBound$ and less than or equal to $rightBound$. **You should write a recursive function for this task.**

Task 5: Add deleteItem function. Add a new function $deleteItem$. The function will delete an existing item from the tree. You must re-structure the tree after deletion. *In case of deletion of a node where both of its children exist, you must choose the next largest node in the tree for replacement.*

You must also satisfy the following requirements:

- Modify main function to test the above operations. Print the tree after every operation.
- You must extend the given code.
- You cannot use any function of C library except $malloc$ and input output functions.
- You cannot use object oriented programming.
- You must *free* unused memory where it is required.
- For any clarification and help, contact your teacher.
- You may be provided an updated and corrected version of this document later if It is required.
- *You must not use other's code. You must not share your code. You must not copy from any other sources such as web, friends, relatives, etc. In all cases, you will earn a 0 and will move closer to getting an "F" grade in the course.*

Assignment designed by Sukarna Barua, *Assistant Professor, CSE, BUET*