Analysis of Algorithms
CS 312 - HBD1
Department of Physics and Computer Science
Medgar Evers College
Exam 3

Instructions:

- **The exam requires completing a set of tasks within 120 minutes.**
- **Write your solutions in the blue book provided.**
- **Use the functions and data structure in the accompanying document when applicable.**
- **Notes are not allowed.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and/or failing to follow any of the rules will result in an automatic zero (0) for the exam.**

TO ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD THE INSTRUCTIONS ABOVE, PRINT YOUR NAME AND THE DATE ON YOUR SUBMISSIONS

Grading

Section	Maximum Points	Points Earned
01	2	
02	2	
03	2	
04	2	
05	2	
06	2	
07	2	
08	2	
09	2	
10	2	
Total	20	

1. **List the following big-oh runtimes in ascending order.**

$\mathbf{O(n)}$	$\mathbf{O(\lceil lg(n) \rceil)}$	$\mathbf{O(1)}$	$\mathbf{O(\lceil n\,lg(n) \rceil)}$	$\mathbf{O(\lceil \sqrt{n} \rceil)}$
$\mathbf{O(\lceil log(n) \rceil)}$	$\mathbf{O(n^2)}$	$\mathbf{O(2^n)}$	$\mathbf{O(\lceil n\,log(n) \rceil)}$	$\mathbf{O(\lceil lg(lg(n)) \rceil)}$

where $\mathbf{lg(n) = log_2(n)}$ and $\mathbf{log(n) = log_{10}(n)}$

2. **Construct the pseudocode for a `Size(S)` algorithm that returns the number of nodes in `S`, where `S` is a pointer to the head of a null-terminated doubly linked list.**

3. **Use the master theorem to find the theta-notation of the function `Fn()` below**

```
Fn(R)
1. if R = null, then
  1. return 0
2. else,
  1. return 1 + Fn(R.left) + Fn(R.right)
```

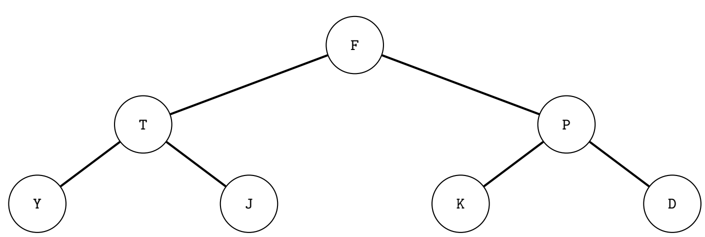
where `R` is a binary tree. You must show work to receive points.

4. **Construct the pseudocode for a `First()` algorithm that takes three integer array parameters as input and returns the maximum value among them. Then, clearly state the input size of the algorithm and determine its Θ-notation time complexity.**

5. **Simulate the execution of the `HeapSort()` algorithm for the input `[48, 96, 92, 62, 31]`. The simulation should start with the original array and then display the state of the array after each swap performed during the sorting process.**

6. **To optimize the `QuickSort()` algorithm, a randomized partition strategy can be used, where the pivot is selected randomly from the subarray instead of always using the last element. Construct the pseudocode for a `RandomizedPartition()` algorithm that implements this approach.**

7. **Perform any two of the tree traversals—`inorder`, `preorder`, or `postorder`—on the given binary tree. For each traversal, write the values on a single line as a list, preceded by the name of the traversal.**



8. **Construct the pseudocode for a `Permutation()` algorithm that takes two strings consisting only of digits as input and returns true if one string is a permutation of the other; otherwise, it returns false. The algorithm must run in $\mathbf{O(n\,lg(n))}$ time or better, where n is the length of the input strings.**

9. **Construct the pseudocode for the `SecondGreatest()` algorithm, which takes a binary search tree (BST) as input and returns the second largest element in the tree.**

10. **The Tower of Hanoi puzzle requires transferring all disks from a source peg to a destination peg such that only the top disk of any peg may be moved at a time, and a disk may be placed only on an empty peg or on top of a larger disk. The disk movement behavior of a peg, therefore, emulates an elementary data structure with an augmented insertion operation.**

    Assuming disks are represented as integers, identify the underlying data structure modeled by a peg in the Tower of Hanoi puzzle. Then, specify the data field used by a *Peg* container class and define the augmented insertion method named `add()`, which takes an integer as input and enforces the Tower of Hanoi constraints.