University of Toronto
Faculty of Arts and Science

Midterm, Summer 2017

CSC263: Data Structure and Analysis
Duration: 60 minutes

First Name: _____   Last Name: _____   Student number: _____

Question	Points	Score
1	40	
2	12	
3	12	
4	16	
5	10	
6	10	
Total:	100	

1. (40 points) **SHORT ANSWER QUESTIONS:**

   (I) **The depths of any two leaves in a max heap differ by at most 1.**
   √ **True**   ◯ False
   Note that the heap is a full tree.

   (II) **A tree with n nodes and the property that the heights of the two children of any node differ by at most 1 has $O(\log n)$ height.** √ **True**   ◯ False
   The mentioned property is the property of AVL Tree. The height of an AVL tree is $O(\log n)$.

   (III) **For any constants $x, y > 1$, we have $n^x = O(y^n)$.** √ **True**   ◯ False In general,
   exponential growth always dominates polynomial growth. To prove, we need to show that $\lim_{n \to \inf} \frac{n^x}{y^n} = 0$. We show that $\frac{n^x}{y^n} < 1/n$.
   $\frac{n^x}{y^n} < 1/n \to n^{x+1} < y^n \to (x+1)\log n < n \log y \to \frac{\log n}{n} < \frac{\log y}{x+1}$.
   Note that for any given x, y there is a value n_0 such that if $n > n_0$, then $\frac{\log n}{n} < \frac{\log y}{x+1}$.

   (IV) **Suppose we have a hash table with m buckets, currently containing n items. Suppose that instead of a linked list, each bucket is implemented as a binary search tree. Give the best asymptotic (big-O) characterization of the worst and best case time complexity of adding an entry to this hash table.**
   √ **Best case $O(1)$, worst case $O(n)$**
   ◯ Best case $O(1)$, worst case $O(\log n)$
   ◯ Best case $O(logn)$, worst case $O(n)$
   ◯ Best case $O(n)$, worst case $O(n)$
   ◯ Best case $O(logn)$, worst case $O(n^2)$
   The best case is when there is one element in the bucket and the worst case is when O(n) elements are in the bucket, note that BST is not always balanced.

   (V) **Inserting an element into an AVL tree with n nodes requires $\Theta(\log n)$ rotations.**
   ◯ True   √ **False**
   Inserting into an AVL tree may look at $O(\log n)$ nodes, but it only needs to perform at most 2 rotations to fix the imbalance. Thus inserting into an AVL tree requires $O(1)$ rotations, which is not $\Theta(logn)$.

   (VI) **For which data structure the order of insertion does not matter, i.e. the resulting data structure is identical regardless of the oder the elements were inserted?**
   ◯ Unsorted sequence
   ◯ Heap
   ◯ AVL tree
   ◯ Hash table with chaining
   √ **None of the above**
   In all the mentioned data structure, different order for insertions could result in a different structure.

   (VII) **Which of the following sorting algorithms in its typical implementation gives best performance when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced).**
   ◯ Quick Sort   √ **Insertion Sort**   ◯ Heap Sort
   See the textbook for the best case running time of each sorting algorithm.

(VIII) **Let T be an AVL tree of height 4. What is the smallest number of entries it can store? Note that a tree with one node (only the root) has the height of zero and stores one element.**

○ 9   ○ 10   ○ 11   √ **12**   ○ 13
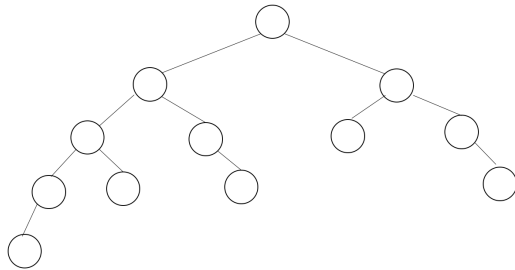
(IX) **Given a hash table T with 25 slots that stores 2000 elements, the load factor α for T is:**

√ **80**   ○ 0.0125   ○ 8000   ○ 1.25

$\alpha = n/m$ where n is the number of elements and m is the number of slots. Therefore $\alpha = 2000/25 = 80$

(X) **In a binary max heap containing n numbers, the smallest element can be found in time**

√ $O(n)$ ○ $O(logn)$ ○ $O(\log\log n)$ ○ $O(1)$

A binary max heap is not sorted, the maximum can be find in $O(1)$, but the minimum can be any leaves of the heap. The number of leaves in a heap is $\lfloor n/2 \rfloor$.

2. (12 points) **GROWTH OF FUNCTIONS:**

   (I) Give the best asymptotic (big-O) and (big-Ω) characterization of the worst case and the best case time complexities of the algorithm DoAgain(A, n)

   Algorithm DoAgain(A, n)
   # Input: Array A of size $n > 1$ storing integers.
   sum $\leftarrow 0$
   for $c = 0$ to n^2 do
       if $A[0] < 0$ then
           for $k = 0$ to $n - 1$ do
               $sum \leftarrow sum + c * A[k]$
   Worst Case: is when $A[0] < 0$: upper bound $O(n^3)$ lower bound $\Omega(n^3)$
   Best Case: is when $A[0] \geq 0$: upper bound $O(n^2)$ lower bound $\Omega(n^2)$

3. (12 points) **HEAP**

   For a max-heap of size n, you will be given the state of the max-heap before an operation and you have to show the state of the max-heap after the operation. Fill in the tables and draw the heap trees before and after the operations

   (I) Insert(16)

   before:

0	1	2	3	4	5	6
32	15	7	6	2	4	3

   after:

0	1	2	3	4	5	6
32	16	7	15	2	4	3	6

   (II) ExtractMax()

   before:

0	1	2	3	4	5	6
32	15	7	6	2	4	3

   after:

0	1	2	3	4	5	6
15	6	7	3	2	4	

4. (16 points) **AVL TREE**

   Perform the following insertions into the given AVL trees and perform the appropriate rotations to rebalance the tree.

   (I) Draw the result when 25 is added to the AVL tree below. Show your work, in particular, identify clearly any rotation performed.

   After inserting 25, the node 30 is left heavy, so a right rotation is needed, but since 23 is also right heavy, a double rotation mudt be performed. WE first needd to do a left rotation on 23.
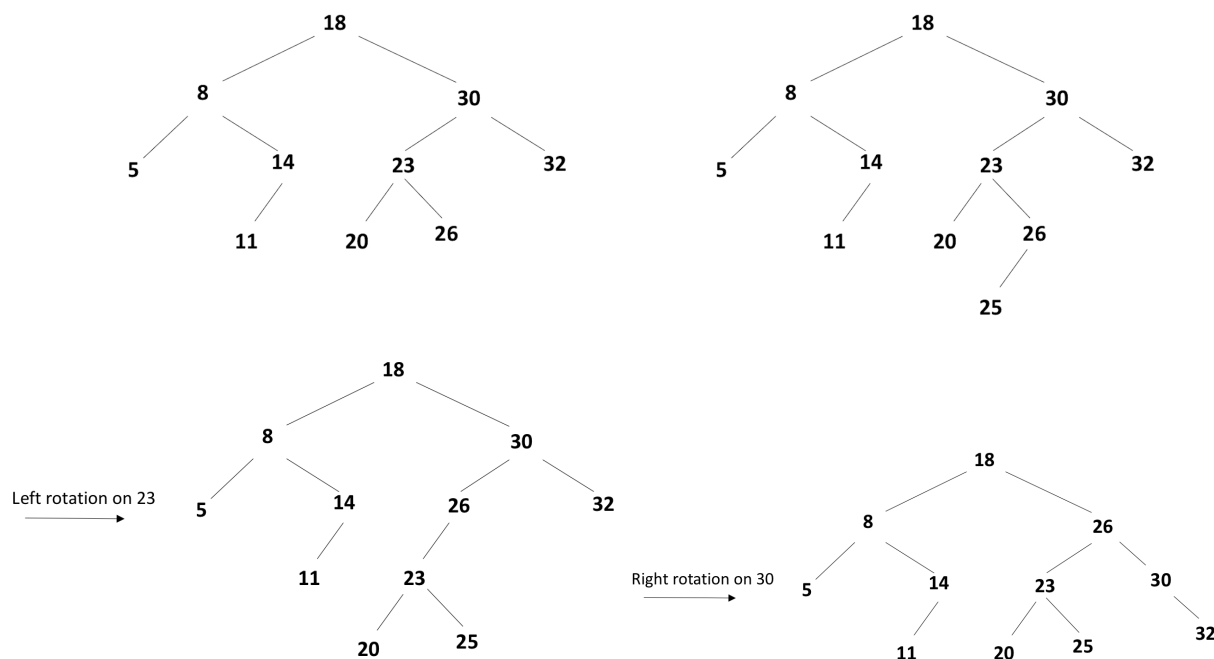
   (II) Draw the result when 18 is deleted from the AVL tree below. Show your work, in particular, identify clearly any rotation performed.

   After deleting 18, there are two possible nodes for the root to be replaced. 14, 23. Since 18 is left heavy, 14 must be chosen. In the resulted 8 is left heavy, so, a right rotation is needed, but since 5 is also right heavy, double rotation must be performed.
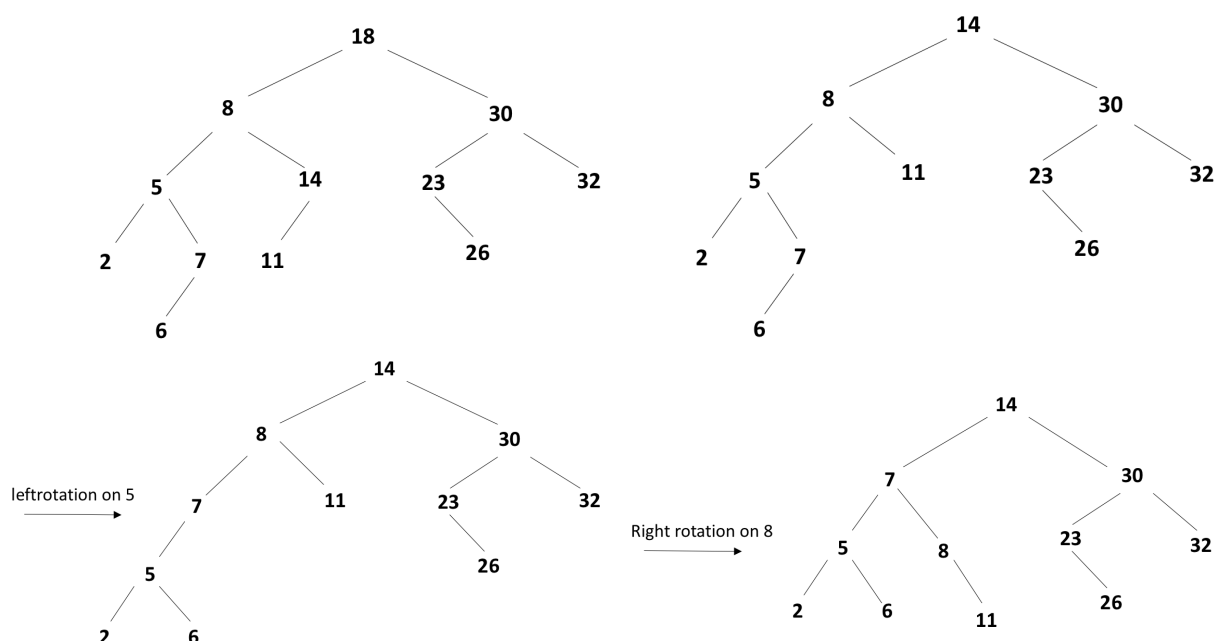
5. (10 points) **HASH TABLE**

   Consider a hash table of size **11** storing entries with integer keys. Suppose the hash function is $h(k) = k \mod 11$. Insert, in the given order, entries with keys $0, 1, 6, 7, 10, 22, 21$ into the hash table using linear probing to resolve collisions. Show all the work.

0	1	2	3	4	5	6	7	8	9	10
0	1	22	21			6	7			10

6. (10 points) **QUICKSORT** Randomized quicksort compares individual pairs of elements but it does not necessarily compare every element to every other element. When the input is the array $[2, 8, 5, 3]$, what is the probability that randomized quicksort compares **2** and **8** directly to each other? Explain your reasoning.

   The probability is $1/2$. If the first pivot is 2 or 8, then 2 and 8 will be compared. If the first pivot is 5 or 3, then 2 and 8 will end up in different sub-arrays and will never be compared to each other.