CS2121/9643 – Assignment 4
due Apr. 5, 2016 (latest to submit: Apr. 8)

1. (10pt) Assume an $n \times n$ matrix A is given, containing only 1's and 0's, such that, in each row, all 1's come before all 0's. Give an $\mathcal{O}(n \log n)$ algorithm to count all 1's in A.

2. (10pt) Let A and B be two sequences of n integers each. Given an integer m, describe an $\mathcal{O}(n \log n)$ algorithm to determine if there is an element a of A and an element b of B such that $m = a + b$.

3. (20pt) Professor X thinks he has discovered a remarkable property of binary search trees. Assume that the search for a key k ends up in a leaf. Consider the sets A, containing the keys to the left of the search path, B, the key on the search path, and C, the keys to the right of the search path. Professor X claims that for any $a \in A, b \in B, c \in C$, we have $a \leq b \leq c$. Give a smallest possible counterexample to this claim.

4. (20pt) What is the minimum number of nodes an AVL tree of height 5 can have? Draw one such AVL tree. Explain why it has the minimum possible number of nodes.

5. (40pt) Write a Python program `compareSort` to compare several sorting algorithms: quick sort, merge sort, and bubble sort. Implement three functions, `quickSort`, `mergeSort`, and `bubbleSort`, the first two recursively. You can use the code from chapters 12 and 5 but you have to fix the bugs in bubble sort and quick sort!!

   Generate a random sequence of 1000 integers (`import random`, use with `random.randint(1, 100000)`), sort it using each of the three algorithms, and record the time (`import time`, use the difference between `time.clock()` taken right before and right after running the sorting function).

   The algorithms have to be compared on *the same* sequences, so, for each algorithm, you need to make a copy and sort the copy. (You can `import numpy` and use the `.copy()` method.) Do not include the time to copy into the sorting time.

   Repeat the whole procedure ten times and output the average time. Here is an example output on my laptop:

```
 quickSort:  0.013 s
 mergeSort:  0.025 s
bubbleSort:  0.748 s
```

Note: Submit your solution on `owl.uwo.ca`: a .py file for question 5 and a .pdf file with all the remaining answers; include also the average times for q5 in the pdf file. Do not submit Python code for questions other than 5!