CSci 4041, Midterm 1 exam

Name:

Student ID number:

Instructions: The time limit is 75 minutes. Please write your answers in the space below. If you need more space, write on the back of the paper. The exam is open book and notes. You may use electronic devices to ONLY look at either an e-book version or electronic notes. You may not use the internet, program/run code or any other outside resources. (If you are typing on your keyboard/input device for anything other than ctrl-F to find words in the e-book or notes, this is probably not acceptable.) **For all questions you must show work**.

Problem 1. (20 points)
Use heapsort (smallest on the left) to sort array A below:
A = [2, 8, 1, 5, 0, 4]

First we need to make A into a heap.

```
          2
      8       1
   5     0   4
```

Heapify 1 and 4:

```
          2
      8       4
   5     0   1
```

Heapify 8 with (5 and 0)... no change:

```
          2
      8       4
   5     0   1
```

Heapify 2 with (8 and 4) swap 2 with 8 then recursively swap 2 with 5:

```
          8
      5       4
   2     0   1
```

Then repeatedly extracti-max and heapify (underlined is ignored part of heap):
[8, 5, 4, 2, 0, 1] extract max
[1, 5, 4, 2, 0, 8] heapify
[5, 2, 4, 1, 0, 8] extract max
[0, 2, 4, 1, 5, 8] heapify
[4, 2, 0, 1, 5, 8] extract max
[1, 2, 0, 4, 5, 8] heapify
[2, 1, 0, 4, 5, 8] extract max
[0, 1, 2, 4, 5, 8] heapify
[0, 1, 2, 4, 5, 8] extract max
... Done

Problem 2. (20 points)
Find the run-time, T(n), for the following recursively defined algorithms (you do not need to simplify you answers):
(1) $T(n) = 10\ T(n/3) + 2^n$

(2) $T(n) = 9\ T(n/3) + n^{2.5}$

(3) $T(n) = T(2n/3\ ) + 5$

(4) $T(n) = 99\ T(99n/100) + n^{99}$

Apply the master's theorem on all of these
(1) 10 T(n/3) acts like n^2.095, 2^n grows faster so O(2^n)

(2) 9 T(n/3) acts like n^2, n^2.5 grows faster so O(n^2.5)

(3) T (2n/3) acts like n^log_1.5(1) = n^0. This grows the same as 5 = O(1). Since they grow the same, we multiply by lg n, so O(1 * lg n) = O(lg n)

(4) 99 T(99n/100) acts like n^log_100/99(99) = n^457.2. This is much bigger than n^99 so O(n^457.2).

Problem 3. (30 points)
You are a merchant traveling with a cart in the preindustrial era. You currently have a wagon of various goods with values stored in an array A. Each of these goods you plan on selling in a different city (for maximum profit), but you can visit cities in any order (the geography of the area makes them all equally distant). When you enter each city to sell the associated good, you must pay a tax of 10% your total current wealth. Describe or write loose pseudo-code for an algorithm to maximize your profit. (In other words what order should you sell all your goods.)

Sample array A:
A = [40, 20, 90, 10, 5]
So item 1 can be sold in city 1 for 40 money. Item 2 can be sold in city 2 for 20 money...

Clarification (was on board): the "wealth" tax refers to the money you are carrying, not the value of the goods.

The greedy choice is to sell the item worth the least, as it will be taxed the most. So you should sort the array A from smallest to largest, then sell the goods in that order (left to right).

Problem 4. (20 points)
Prove f(n) = $\ln(n^3)$ is O($\ln(n^2)$). (Yes, there is a 3 on the left and a 2 on the right.)
Math fact: $\ln(a^b) = a*\ln(b)$

Correction: Math fact was terrible... but even if you assume this, you should be able to prove the desired fact.

Proof with incorrect fact:
$\ln(n^3) = n \ln 3$. Let $C = \ln 4 / \ln 2 > \ln 3 / \ln 2$. Then...
RHS = $C * n * \ln 2 = n * \ln 4 / \ln 2 * \ln 2 = n * \ln 4 \geq n * \ln 3$ for all $n \geq 0$.
Thus $C = \ln 4 / \ln 2$, $n_0 = 0$

Proof with correct fact (i.e. $\ln(a^b) = b \ln(a)$):
$\ln(n^3) = 3 \ln n$. Let $C = 2$. Then ..
RHS = $C * 2 * \ln n = 4 \ln n \geq 3 \ln n$, for all $n \geq 1$
Thus $C = 2$, $n_0 = 1$.

Problem 5. (20 points)
Find the worst case and average runtime of the following pseudo code.

```
MergeTastic(A)        // Comment: A is a linked list
while A.size > 1
        i = argmin(A)
        iVal = A[i]
        A.remove(i)
        j = argmin(A)
        jVal = A[j]
        A.remove(j)
        A.addToStart(iVal + jVal) //Comment: (iVal+jVal) is the value being added to the start of the list
```

For a linked list: argmin() = O(n), A[i] = O(i), remove(i) = O(i) or O(1) (depending on whether you consider that we need to research to get to i. Both will have the same runtime), and addToStart() = O(1).

So the amount of work done in the while loop is:
2 * argmin() + A[i] + A[j] + remove(i) + remove(j) + addToStart.

Worst-case is [i] and [j] are size of A. Let this be "n". Then the run time is:
2 * O(n) + O(n) + O(n) + O(n) + O(n) + O(1) = O(n) within the loop
The loop then runs "n" times, which makes the total runtime = O(n^2)

Average case is [i] and [j] are n/2. Then the run time is:
2 * O(n) + O(n/2) + O(n/2) + O(n/2) + O(n/2) + O(1) = O(n) within the loop
The loop then runs "n" times, which makes the total runtime = O(n^2)