You may work in pairs for this assignment. If you choose to work with a partner, make sure only one of you submits a solution, and you paste a copy of the Partners Template that contains the names and PIDs of both students at the beginning of the file you submit.

You will submit your solution to this assignment to the Curator System (as `HW03`). Your solution must be either a plain text file (e.g., NotePad++) or a typed MS Word document; submissions in other formats will not be graded.

Partial credit will only be given if you show relevant work.

In all questions about complexity, functions are assumed to be nonnegative.

1. [36 points] The analysis of a certain algorithm leads to the following complexity function (for the average case):

$$T(N) = 3N\log^2(N) + 5N^2\log(N) + 2N\log(N) + 23$$

Several computer science students offer their conclusions about the algorithm (quoted below). For each conclusion, state whether it is correct, or incorrect, or could be either correct or incorrect, based on the given information, and give a brief justification of your answer; feel free to cite any relevant theorems from the course notes.

It helps to know what category T(N) belongs to:

$$\lim_{N\to\infty}\frac{3N\log^2(N)+5N^2\log(N)+2N\log(N)+23}{N^2\log(N)} = \lim_{N\to\infty}\left(\frac{3\log(N)}{N}+5+\frac{2}{N}+\frac{23}{N^2\log(N)}\right)=5$$

So, T(N) is $\Theta(N^2\log(N))$.

Aside from that, we need to think about how the average case tells us about the best and worst cases:

- **the average case serves as a lower bound for the worst case; the worst case could be MUCH worse than the average case, but it cannot be better, although the average and worst cases could be the same**
- **the average case serves as an upper bound for the best case; the best case could be MUCH better than the average case, but it cannot be worse, although the average and best cases could be the same**

On average…

a) … the algorithm is $O(N^3)$.

Correct. (But it's a strict upper bound.)
On average, T(N) is $\Theta(N^2\log(N))$, so it's also $O(N^2\log(N))$. And, $N^2\log(N)$ is $O(N^3)$. You can show that by using the limit theorem, but a simple, intuitive argument is that $N^3 = N^2*N > N^2\log(N)$, since $N > \log(N)$.

b) … the algorithm is $\Omega(N^3)$.

Incorrect.
The proof is actually part a). If N3 is a strict upper bound on the average case, it cannot also be a lower bound on the average case.

c) … the algorithm is $\Omega(N^2 \log N)$.

Correct.
Since T(N) is $\Theta(N^2\log(N))$, by definition T(N) is $\Omega(N^2\log(N))$.

In the worst case…

d) … the algorithm is $\Theta(N^2 \log N)$.

Could be correct or incorrect.
The worst case has the best case as a lower bound, so the worst case is surely $\Omega(N^2\log(N))$. It's possible the worst case is also $O(N^2\log(N))$, but we don't have any information that would tell us whether that's true.

e) … the algorithm is $O(N^3)$.

Could be correct or incorrect.
We know the average case is $O(N^3)$, from an earlier question. And, we know the worst case is $\Omega(N^2\log(N))$. So, the worst case could be no worse than N^3, but we don't have any information that would tell us whether that's true.

f) … the algorithm is $\Omega(N^2 \log N)$.

Correct.
The average case is always a lower bound for the worst case.

In the best case…

g) … the algorithm is $O(N \log N)$.

Could be correct or incorrect.
The best case has the average case as an upper bound, so we know the best case is $O(N^2\log(N))$. And, it's clear that N log(N) is strictly $O(N^2\log(N))$. So, if the best case is much better than $N^2\log(N)$, then it could be $O(O(N\log(N))$, but we have no information that would tell us whether that's true.

h) … the algorithm is $O(N^2 \log(N))$.

Correct.
The average case is always an upper bound for the best case.

i) … the algorithm is $\Omega(1)$.

Correct.
Surely the best case has some positive cost, and every positive function is $\Omega(1)$.

2. [24 points] Suppose that an algorithm takes 30 seconds for an input of 2^{16} elements (with some particular, but unspecified speed in instructions per second). Estimate how long the same algorithm, running on the same hardware, would take if the input contained 2^{24} elements, and that the algorithm's complexity function is:

a) $\Theta(N)$
b) $\Theta(\log N)$
c) $\Theta(N \log N)$
d) $\Theta(N^2)$

Assume that the low-order terms of the complexity functions are insignificant, and state your answers in the form HH:MM:SS.S (hours, minutes, seconds, tenths of a second). Be sure to show supporting work.

It helps to start with the following observation. Since the algorithm takes 30 seconds for an input of size 2^{16}, we know that if the hardware can execute S instructions per second $T(2^{16}) / S =$ 30 seconds.

a) Suppose T(N) = N (we can ignore low-order terms). Then

$$T(2^{24}) / S = 2^{24} / S = 2^8 * (2^{16} / S) = 256 * 30 \text{ seconds} = 128 \text{ minutes}$$

So, the running time would be about 02:08:00.

b) Suppose T(N) = log N. Then

$$T(2^{16}) / S = \log(2^{16}) / S = 16 / S$$

and

$$T(2^{24}) / S = \log(2^{24}) / S = 24 / S = 1.5 * (16 / S) = 1.5 * 30 \text{ seconds} = 45 \text{ seconds}$$

So, the running time would be about 00:00:45.

c) Suppose T(N) = N log N. Then

$$T(2^{16}) / S = 2^{16} \log(2^{16}) / S = 16 * 2^{16} / S = 2^{20} / S$$

and

$$T(2^{24}) / S = 2^{24} \log(2^{24}) / S = 24 * 2^{24} / S = (24 * 2^4) * (2^{20} / S)$$
$$= 384 * 30 \text{ seconds} = 192 \text{ minutes}$$

So, the running time would be about 03:12:00.

d) Suppose $T(N) = N^2$. Then

$$T(2^{16}) / S = 2^{32} / S$$

and

$$T(2^{24}) / S = 2^{48} / S = 2^{16} * 2^{32} / S = 2^{16} * 30 \text{ seconds} = 2^{15} \text{ minutes}$$

So, the running time would be 22:18:08:00 (dd:hh:mm:ss) or 546:08:00 (hh:mm:ss).

3. [18 points] Use theorems from the course notes to solve the following problems. Show work to support your conclusions.

a) Find the "one-term" function g(n) such that

$$f(n) = 17n^{5/2} + 3n^2 \log n + 1000 \text{ is } \Theta(\text{g(n)})$$

From the notes, I will guess that $n^{5/2}$ log n will be dominant, but we need to check:

$$\underset{n\to\infty}{\text{limit}} \frac{17n^{5/2} + 3n^2 \log n + 1000}{n^{5/2}} = \underset{n\to\infty}{\text{limit}} \left(17 + \frac{3\log n}{n^{1/2}} + \frac{1000}{n^{5/2}} \right) = 17 + \underset{n\to\infty}{\text{limit}} \left(\frac{3/n \ln 2}{1/2 * n^{-1/2}} \right)$$

$$= 17 + \underset{n\to\infty}{\text{limit}} \left(\frac{6}{n^{1/2} \ln 2} \right) = 17$$

So, f(n) is $\Theta(n^{5/2})$.

b) Find the "simplest one-term" function g(n) such that

$$f(n) = 5n\log^2(n) + 8n\log(\text{n}^2) \text{ is } \Theta(\text{g(n)})$$

Considering laws of logarithms, I think the first term is dominant, but I must show that:

$$\underset{n\to\infty}{\text{limit}} \frac{5n\log^2(n) + 8n\log(\text{n}^2)}{n\log^2(n)} = \underset{n\to\infty}{\text{limit}} \left(5 + \frac{16n\log(n)}{n\log^2(n)} \right) = \underset{n\to\infty}{\text{limit}} \left(\frac{16}{\log(n)} \right) + 5 = 5$$

So, f(n) is $\Theta(n \log^2 n)$.

c) Find the "simplest one-term" function g(n) such that

$$f(n) = 3^n + n2^n \text{ is } \Theta(\text{g(n)})$$

Here, the first term is clearly (intuitively) dominant, but we must show that:

$$\underset{n\to\infty}{\text{limit}} \frac{3^n + n2^n}{3^n} = \underset{n\to\infty}{\text{limit}} \left(1 + \frac{n2^n}{3^n} \right) = \underset{n\to\infty}{\text{limit}} \left(1 + \frac{n}{1.5^n} \right) = 1 + \underset{n\to\infty}{\text{limit}} \left(\frac{1}{1.5^n \ln 1.5} \right) = 1$$

So, f(n) is $\Theta(3^n)$.

4. [12 points] Using the counting rules from the course notes, find the exact-count complexity function T(n) for the following algorithm. Show details of your analysis, and simplify your answer. In simplifying, you may discard the floor notation, if relevant.

```
x = 100;                               // 1
y = 0;                                 // 1
for (r = 1; r <= n; r = 2*r) {         // 1 before, 3 per pass, 1 to exit
   x = x + r;                          // 2 per pass
   for (c = 2; c < n; c++) {           // 1 before, 2 per pass, 1 to exit
      if ( x > y / c )                 // 2
         y = y + r / c;                // 3
   }
}
```

$$T(n) = 1 + 1 + 1 + \sum_{p=1}^{1+\log n}\left(3 + 2 + 1 + \sum_{c=2}^{n-1}(2 + 2 + 3) + 1\right) + 1$$

$$= \sum_{p=1}^{1+\log n}\left(\sum_{c=2}^{n-1} 7 + 7\right) + 4 = \sum_{p=1}^{1+\log n}(7n - 7) + 4$$

$$= 7n\log n + 7n - 7\log n - 3$$

5. [10 points] Use the definition of Θ to prove that, if *b* is an arbitrary positive integer, then

$$(n+1)^b \text{ is } \Theta(n^b)$$

Hint: One part is trivial. The other part is not trivial. You'll need to use some clever algebra. This result is also true if we drop the assumption that *b* is an integer; you might find this to be easier if you ignore that assumption as well.

Warning: a proof using any theorems, including the limit theorem, will receive NO credit.

This will be discussed in class.