HW 1 SOLUTIONS

Due 23:59 January 11, 2019

CS ID 1: ___________

CS ID 2: ___________

Instructions:

1. Do not change the problem statements we are giving you. Simply add your solutions by editing this latex document.
2. Take as much space as you need for each problem. You'll tell us where your solutions are when you submit your paper to gradescope.
3. Export the completed assignment as a PDF file for upload to gradescope.
4. On gradescope, upload only **one** copy per partnership. (Instructions for uploading to gradescope will be posted on the HW1 page of the course website.)

Academic Conduct: I certify that my assignment follows the academic conduct rules for of CPSC 221 as outlined on the course website. As part of those rules, when collaborating with anyone outside my group, (1) I and my collaborators took no record but names away, and (2) after a suitable break, my group created the assignment I am submitting without help from anyone other than the course staff.

1. (2 points) Using 140 characters or less, post a synopsis of your favorite movie to the course piazza space under the "HW1 tell me something!" notice, so that your post is visible to everyone in the class, and tagged by #HW1num1. Also, use Piazza's code-formatting tools to write a *private* post to course staff that includes at least 5 lines of code. It can be code of your own or from a favorite project—it doesn't even have to be syntactically correct—but it must be formatted as a code block in your post, and also include the tag #HW1num1. (Hint: Check http://support.piazza.com/customer/portal/articles/1774756-code-blocking). Finally, please write the 2 post numbers corresponding to your posts here:

Favorite Movie Post (Public) number:	
Formatted Code Post (Private) number:	

2. (16 points) In this problem, you will be a math detective! Some of the symbols and functions may not be familiar to you, but with a little digging, reading, and observing, you'll be able to figure them out. Your task is to simplify each of the following expressions as much as possible, **without using an calculator (either hardware or software)**. Do not approximate. Express all rational numbers as improper fractions. You may assume that n is an integer greater than 1. Show your work in the space provided, and write your final answer in the box.

   (a) i. $\sum_{k=1}^{n} k2^{k+1} - \sum_{k=1}^{n} k2^{k}$

Answer for (a.i):	$(n-1)2^{n+1}+2$

$$
\begin{aligned}
&= 0 \times 2^1 + \sum_{k=1}^{n-1} k2^{k+1} + n2^{n+1} - \sum_{k=1}^{n} k2^k \\
&= \sum_{k=0}^{n-1} k2^{k+1} - \sum_{k=1}^{n} k2^k + n2^{n+1} \\
&= \sum_{k=1}^{n} (k-1)2^k - \sum_{k=1}^{n} k2^k + n2^{n+1} \\
&= \sum_{k=1}^{n} -2^k + n2^{n+1} \\
&= -(2^{n+1} - 1 - 1) + n2^{n+1} \\
&= (n-1)2^{n+1} + 2
\end{aligned}
$$

ii. $\displaystyle\prod_{i=1}^{n-1} \frac{2i}{n-i}$

Answer for (a.ii):	2^{n-1}

$$\begin{aligned}\prod_{i=1}^{n-1} \frac{2i}{n-i} &= 2^{n-1}\frac{\prod_{i=1}^{n-1} i}{\prod_{i=1}^{n-1} n-i}\\ &= 2^{n-1}\frac{\prod_{i=1}^{n-1} i}{\prod_{j=1}^{n-1} j} \qquad \text{(for } j = n-i\text{)}\\ &= 2^{n-1}\end{aligned}$$

(b) i. $15^{333} \bmod 2$

Answer for (b.i):	1

Use the fact that $((a \mod c) \cdot (b \mod c)) \mod c = a \cdot b \mod c$, or its implication that multiplying two odd numbers results in an odd number. To see $((a \mod c) \cdot (b \mod c)) \mod c = a \cdot b \mod c$, let $x = a \mod c$ and $y = b \mod c$, that is, $a = pc + x$ and $b = qc + y$ where p and q are integers. So $a\dot{b} = (pc + x)(qc + y) = pqc^2 + pyc + qxc + xy$ and, since the first three terms have a factor of c, $a\dot{b} \mod c = xy = (a \mod c) \cdot (b \mod c)$.

ii. $32^{333} \bmod 15$

Answer for (b.ii):	2

First, $32^{333} = 2^{5 \cdot 333} = 16^{5 \cdot 333/4} = 16^{416+1/4} = 16^{416} \cdot 2$. Then, using the fact above, $16^{416} \cdot 2 \mod 15 = (16^{416} \mod 15)(2 \mod 15) \mod 15 = (1 \mod 15)(2 \mod 15) = 2$.

(c) i. $\displaystyle\sum_{r=0}^{n}\left(\frac{1}{n}\right)^r$

Answer for (c.i):	$\dfrac{1-\left(\frac{1}{n}\right)^{n+1}}{1-\frac{1}{n}}$

This is the sum of a geometric series with common ratio $1/n$. If S is the sum, then $S - S/n = 1 - (1/n)^{n+1}$ (since all terms with corresponding powers of $1/n$ cancel), so $S = \frac{1-(1/n)^{n+1}}{1-1/n}$.

ii. $\displaystyle\sum_{r=3}^{\infty}\left(\frac{2}{3}\right)^r$

Answer for (c.ii):	8/9

Another geometric series. Since $2/3 < 1$, the sum of the series converges. If S is the sum, then $S - 2S/3 = (2/3)^3$, so $S = (2/3)^3/(1 - 2/3) = 8/9$.

(d) i. $4^{(\log_2 n)/2}$

Answer for (d.i):	n

$$4^{(\log_2 n)/2} = \sqrt{4}^{\log_2 n} = 2^{\log_2 n} = n.$$

ii. $\dfrac{\log_7 1024}{\log_7 32} - \dfrac{\log_3 1024}{\log_3 32}$

Answer for (d.ii):	0

Both terms express $\log_{32} 1024$.

iii. $\dfrac{\log_2 n}{\log_{32} n}$

Answer for (d.iii):	5

$$\frac{\log_2 n}{\log_{32} n} = \frac{\log_2 n}{(\log_2 n)/(\log_2 32)} = \log_2 32 = 5$$

3. (12 points)

(a) (3 points) Find a closed form expression for $f(n) = \sum_{j=1}^{n}(2j-1), \forall n \geq \underline{\mathbf{1}}$ (fill in the blank with the smallest reasonable value!), and use induction to prove the formula is correct.

Formula:	$f(n) = n^2$

SOLUTION: This is a straightforward proof by induction. We prove:

$$\forall n \geq 1, \sum_{k=1}^{n}(2k-1) = n^2.$$

Consider an arbitrary integer $n \geq 1$.

If $n = 1$ (base case), the sum gives $(2n-1) = 1$ and the closed form gives $n^2 = 1$, so the statement holds.

A helpful assumption (inductive hypothesis):

$$\forall 1 \leq j < n, \sum_{k=1}^{j}(2k-1) = j^2.$$

Let's simply solve the sum:

$$\begin{aligned}
\textstyle\sum_{k=1}^{n} &= \textstyle\sum_{k=1}^{n-1}(2k-1) + 2n - 1 && \text{(peel off last term)}\\
&= (n-1)^2 + 2n - 1 && \text{(by inductive hypothesis)}\\
&= (n^2 - 2n + 1) + 2n - 1 && \text{(by algebra)}\\
&= n^2 && \text{(by algebra)}
\end{aligned}$$

(b) (2 points) Find a closed form expression for

$$\frac{1 + 3 + 5 + \ldots + (2k-1)}{(2k+1) + (2k+3) + \ldots + (4k-1)}.$$

Formula:	$\frac{1}{3}$

(c) Give a rigorous, direct proof that your solution to part (b) is correct.

i. (2 points) Express the fraction using a new variable that quantifies the number of terms in the numerator and in the denominator. Be sure to bound that variable.

SOLUTION:

$$\frac{\sum_{j=1}^{n}(2j-1)}{\sum_{j=n+1}^{2n}(2j-1)}, n \geq 1$$

Formula:	see expression in text

ii. (5 points) Complete the proof. A concise solution will apply the result from part (a) three times.

SOLUTION:

As usual, there are alternative approaches to this problem. I'll show one. There is at least one other that is equally elegant, but which applies the result from part (a) only twice.

We prove that $\forall n \geq 1$,

$$\frac{\sum_{j=1}^{n}(2j-1)}{\sum_{j=n+1}^{2n}(2j-1)} = \frac{1}{3}.$$

The numerator is n^2 by direct application of part (a). The denominator can be re-expressed as:

$$\sum_{j=n+1}^{2n}(2j-1) = \sum_{j=1}^{2n}(2j-1) - \sum_{j=1}^{n}(2j-1)$$

After two more applications of the result from part (a) we have:

$$\frac{n^2}{(2n)^2 - n^2} = \frac{1}{3}.$$

4. (8 points) Prove that $f(n) \in O(g(n))$ if and only if $g(n) \in \Omega(f(n))$.

SOLUTION: $f(n) \in O(g(n))$ implies (by the definition of O) there are positive constants $c \in \mathbb{R}$ (a real number) and $n_0 \in \mathbb{Z}$ (an integer) such that $f(n) \leq cg(n)$ for all $n \geq n_0$. Equivalently, $g(n) \geq \frac{1}{c}f(n)$ for all $n \geq n_0$. This implies (by the definition of Ω) that $g(n) \in \Omega(f(n))$, since $1/c$ is a positive real number and n_0 is a positive integer, such that $g(n) \geq \frac{1}{c}f(n)$ for all $n \geq n_0$.

$g(n) \in \Omega(f(n))$ implies (by the definition of Ω) there are positive constants $c \in \mathbb{R}$ (a real number) and $n_0 \in \mathbb{Z}$ (an integer) such that $g(n) \geq cf(n)$ for all $n \geq n_0$. Equivalently, $f(n) \leq \frac{1}{c}g(n)$ for all $n \geq n_0$. This implies (by the definition of O) that $f(n) \in O(g(n))$, since $1/c$ is a positive real number and n_0 is a positive integer, such that $f(n) \leq \frac{1}{c}g(n)$ for all $n \geq n_0$.

5. (8 points) Indicate for each of the following pairs of expressions $(f(n), g(n))$, whether $f(n)$ is O, Ω, or Θ of $g(n)$. Prove your answers. Note, if you choose O or Ω, you must also show that the relationship is not Θ. You may use the result from problem 4, if you find it useful.

   (a) $f(n) = n \log(n^2 + 1) + n^2 \log n$, and $g(n) = n^2$.

Answer for (a):	$f(n) \in \Omega(g(n))$

SOLUTION: From problem 4, we know that $f(n) \in \Omega(g(n))$ if and only if $g(n) \in O(f(n))$. We show the latter.

Show that $\exists c > 0, n_0 > 0$, such that $\forall n \geq n_0, n^2 \leq c(n \log(n^2 + 1) + n^2 \log n)$.

Let $c = 1, n_0 = 2$. Then,

$$\begin{aligned} n^2 &\leq n^2 \log n && \text{since } n \geq 2 \text{ implies } \log n \geq 1. \\ &\leq n \log(n^2 + 1) + n^2 \log n && \text{since } n \log(n^2 + 1) \geq 0 \end{aligned}$$

We next show that $f(n) \notin O(g(n))$ (so $f(n)$ is not $\Theta(g(n))$). We'll prove this by contradiction. Let's assume (in order to derive a contradiction) that $f(n) \in O(g(n)$. By the definition of O, there is a positive real number c and a positive integer n_0 so that $f(n) \leq cg(n)$ for all $n \geq n_0$. Substituting our functions, this implies:

$$n \log(n^2 + 1) + n^2 \log n \leq cn^2 \qquad \forall n \geq n_0$$

which implies

$$n^2 \log n \leq cn^2 \qquad \forall n \geq n_0$$

which implies

$$\log n \leq c \qquad \forall n \geq n_0.$$

However, if we choose $n > \max\{10^c, n_0\}$, then $\log n > c$ for some $n \geq n_0$, which is a contradiction, so $f(n) \notin O(g(n))$.

(b) $f(n) = n^2$, and $g(n) = \frac{1}{2}n^3 - 2n^2 - 3n - 17$.

Answer for (b):	$f(n) \in O(g(n))$

SOLUTION:
Let $c = 4$, and $n_0 = 88$. Show that $\forall n \geq 88, n^2 \leq 4(\frac{1}{2}n^3 - 2n^2 - 3n - 17)$.
(Note that this problem would be easy if there were no negative terms. There are a couple different approaches to solving the problem. We'll show one.)
Consider an arbitrary $n \geq 88$. Observe:

$$n^2 \leq n^3 + B(n), \quad \text{if } B(n) \geq 0.$$

Let $B(n) = n^3 - 8n^2 - 12n - 68$. Then, to show $B(n) \geq 0$ we just need to show that:

$$n^3 \geq 8n^2 + 12n + 68$$

To that end:

$$\begin{aligned} n^3 &\geq 88n^2, && \text{(since } n \geq 88\text{)} \\ &= 8n^2 + 12n^2 + 68n^2, && \text{(by algebra)} \\ &\geq 8n^2 + 12n + 68, && \text{(since } n \geq 1\text{).} \end{aligned}$$

Finally, we put these pieces together:

$$\begin{aligned} n^2 &\leq n^3 + n^3 - 8n^2 - 12n - 68, && \text{(by argument above.)} \\ &= 4(\tfrac{1}{2}n^3 - 2n^2 - 3n - 17), && \text{(by algebra)} \end{aligned}$$

■
To show that $g(n) \notin O(f(n))$, we use a proof by contradiction.
From the definition of big-O, we know there exist positive constants c and n_0 so that

$$\frac{n^3}{2} - 2n^2 - 3n - 17 \leq cn^2, \forall n \geq n_0.$$

Equivalently, we have

$$n^3 \leq 2cn^2 + 4n^2 + 6n + 34, \forall n \geq n_0.$$

Consider any $n > \max\{n_0, 2c + 44, 1\}$. Then

$$\begin{aligned} n^3 &> (2c + 44)n^2 && \text{(by choice of n)} \\ &= 2cn^2 + 4n^2 + 6n^2 + 34n^2 && \text{(by algebra)} \\ &\geq 2cn^2 + 4n^2 + 6n + 34 && \text{(since } n \geq 1\text{)} \end{aligned}$$

This is a contradiction, so our assumption must have been incorrect, and $g(n) \notin O(f(n))$.■

6. (16 points) For each C++ function below, give the tightest asymptotic upper bound that you can determine for the function's **runtime**, in terms of the input parameter. Where prompted, also compute the return value of the function. For all function calls, assume that the input parameter $n \geq 2$.

(a)

```
int raspberryscone(int n) {
     int r;
   int s = 0;
   for (int q = 0; q < n; q++)
      s = s + q;
   for (r = s; r > 2; r--)
     s--;
   return r * s;
}
```

If $n = 2$ then $s = 1$ at the start of the second for-loop. That for-loop only sets r to equal s (and doesn't decrement r or s), and the function returns 1. If $n > 2$, the function returns 4, since s (and thus r) starts the second for-loop with value at least 2, and the for-loop terminates with $s = r = 2$.

Return value for (a):	$\begin{cases} 1 & \text{if } n = 2 \\ 4 & \text{otherwise} \end{cases}$

The first for-loop takes time n to create $s = n(n-1)/2$ which the second for-loop takes time about $n(n-1)/4$ to decrement.

Running time of (a):	$O(n^2)$

(b)

```
int peanutbuttercheesecake(int n) {
   int i = 0;
   int j = 4 * n;
   for (int k = j * j; k > 1; k = k / 2) {
      i++;
   }
   return i * i / 4;
}
```

The loop executes $\lfloor \lg((4n)^2) \rfloor = \lfloor 2 \lg n + 4 \rfloor$ times, so the return value is $\lfloor 2 \lg n + 4 \rfloor^2/4$.

Return value for (b):	$\lfloor 2 \lg n + 4 \rfloor^2/4$

Running time of (b):	$O(\log n)$

(c)

```
int almondcaramelbrittle(int n) {
   int c = 0;
   int s;
   for (s = 1; s < n * n * n * n; s = s * n) {
         c++;
   }
   return c * s;
}
```

Loop executes 4 times, terminating with $c = 4$ and $s = n^4$.

Return value for (c):	$4n^4$

Running time of (c):	$O(1)$

(d)

```
void vanillacustardblueberrytorte(int n) {
   int j = 1;

   for (int k = 0; k < raspberryscone(n/2) * almondcaramelbrittle(n); k++)
         j = 2 * j;

   for (int m = j; m > 1; m = m / 2)
         cout << "I am having so much fun with asymptotics!" << endl;
}
```

Since the return value of `raspberryscone(n/2)` is 4 (or 1 if $n \leq 5$) and the return value of `almondcaramelbrittle(n)` is $4n^4$, their product p is in $\Theta(n^4)$. The first for-loop executes p times and sets $j = 2^p$ by the end. The second for-loop executes $\lg(j) = p$ times. The running time of the second loop (since it takes constant time per iteration) is dominated by that of the first loop. So we only need to calculate a tight asymptotic upper bound on the running time of the first loop.

The first loop's running time depends on whether the calls to calculate
`raspberryscone(n/2) * almondcaramelbrittle(n)`
execute on **every iteration** (the default C++ behaviour) or **only once** (what an optimizing compiler might do). Let $T_{\text{ras}}(n/2)$ and $T_{\text{alm}}(n)$ be the running times of these two calls. Runtime of first loop is,

every iteration case: $O(p(T_{\text{ras}}(n/2) + T_{\text{alm}}(n))) = O(p((n/2)^2 + 1)) = O(n^6)$.

only once case: $O(T_{\text{ras}}(n/2) + T_{\text{alm}}(n) + p) = O((n/2)^2 + 1 + p) = O(n^4)$.

Running time of (d):	$O(n^6)$ or $O(n^4)$

(e)

```
int pirouline(int n) {
    int j = 0;
    for (int k = 0; k < raspberryscone(n) * n; k++)
        j = j + 2;
   return peanutbuttercheesecake(almondcaramelbrittle(j));
}
```

The loop executes $4n$ times (let's assume $n > 2$) to produce $j = 8n$. The loop's running time depends on whether the call to calculate `raspberryscone(n)` executes on **every iteration** (the default C++ behaviour) or **only once** (what an optimizing compiler might do). Runtime of the loop is,

every iteration case: $O(4nT_{\text{ras}}(n)) = O(4n(n^2)) = O(n^3)$.

only once case: $O(4n + T_{\text{ras}}(n)) = O(4n + n^2) = O(n^2)$.

The call to `almondcaramelbrittle(j)` takes $O(1)$ time to produce $j^4 = (8n)^4$. The call to `peanutbuttercheesecake` then takes $O(\log(j^4)) = O(\log n)$ time. This is dominated by the loop's runtime in both cases.

Running time of (e):	$O(n^3)$ or $O(n^2)$

Blank sheet for extra work.