Problem 1

Show that $\sum_{i=1}^{N} i^3 = (\sum_{i=1}^{N} i)^2$ using induction.

Base case:

Let N=4

Left side:

$1^3 + 2^3 + 3^3 + 4^3$

$= 100$

Right side:

$(1+2+3+4)^2$

$= (10)^2$

$= 100$

The left and right side are equal, so the formula is true when N=4

Inductive Hypothesis:

It is true that $\sum_{i=1}^{N+1} i^3 = (\sum_{i=1}^{N+1} i)^2$

Proof:

Left side:

$\sum_{i=1}^{N+1} i^3 = \left(\sum_{i=1}^{N} i^3\right) + (N+1)^3$

Right side:

$(\sum_{i=1}^{N+1} i)^2 = \left((\sum_{i=1}^{N} i) + (N+1)\right)^2$

$= (\sum_{i=1}^{N} i)^2 + 2(\sum_{i=1}^{N} i)(N+1) + (N+1)^2$

Both sides together:

$\left(\sum_{i=1}^{N} i^3\right) + (N+1)^3 = (\sum_{i=1}^{N} i)^2 + 2(\sum_{i=1}^{N} i)(N+1) + (N+1)^2$

$(N+1)^3 = 2(\sum_{i=1}^{N} i)(N+1) + (N+1)^2$

$(N+1)^2 = 2(\sum_{i=1}^{N} i) + (N+1)$

$(N+1)^2 = 2\left(\frac{\mathrm{N(N+1)}}{2}\right) + (N+1)$, theorem on textbook page 5

$(N+1)^2 = N(N+1) + (N+1)$

`

$N + 1 = N + 1$

Since both sides are equal, they are indeed equivalent.

`

Problem 2

Recursive method for number of 1s in binary representation of N

Assuming N is a non-negative integer,

```
private static int binaryOnes(int n)
{
   if (n == 1 || n == 0)
      return n;
   else if (n % 2 == 0)
      return binaryOnes(n / 2); // even
   else
      return binaryOnes(n / 2) + 1; // odd
}
```

(Scratch work)

10 => 1010

5 = > 101

6=>110

7 => 111

14 =>1110

28 => 11100

30 => 11110

31 => 11111

31/2=15 => 1111

f(15)+1

15 => 1111

`

Problem 3

Show that $\sum_{i=1}^{N} \mathrm{i} \times \mathrm{i}! = (N+1)! - 1$ using induction.

Base:

Let N=2

Left side:

$\sum_{i=1}^{2} \mathrm{i} \times \mathrm{i}!$

$= (1 \times 1!) + (2 \times 2!)$

$= (1) + (4)$

$= 5$

Right side:

$(2+1)! - 1$

$= 3! - 1$

$= 6 - 1$

$= 5$

The left and right sides are equal, so the base case is true.

Inductive Hypothesis:

It is true that $\sum_{i=1}^{N+1} \mathrm{i} \times \mathrm{i}! = ((N+1)+1)! - 1$

Proof

Right side:

$\sum_{i=1}^{N+1} \mathrm{i} \times \mathrm{i}!$

$= (\sum_{i=1}^{N} \mathrm{i} \times \mathrm{i}!) + (N+1)(N+1)!$

Left side:

$((N+1)+1)! - 1$

$= (N+2)! - 1$

Both sides together:

$(\sum_{i=1}^{N} \mathrm{i} \times \mathrm{i}!) + (N+1)(N+1)! = (N+2)! - 1$

$(\sum_{i=1}^{N} \mathrm{i} \times \mathrm{i}!) + (N+1)(N+1)! = (N+1)! \times (N+2) - 1$

`

$(\sum_{i=1}^{N} \mathrm{i} \times \mathrm{i}!) = (N+2)(N+1)! - (N+1)(N+1)! - 1$

$(\sum_{i=1}^{N} \mathrm{i} \times \mathrm{i}!) = \big((N+2) - (N+1)\big)(N+1)! - 1$

$(\sum_{i=1}^{N} \mathrm{i} \times \mathrm{i}!) = (1)(N+1)! - 1$

$(\sum_{i=1}^{N} \mathrm{i} \times \mathrm{i}!) = (N+1)! - 1$

The function has been simplified, and results in the original function. Therefore, the function must be true.

`

Problem 4

List the functions from lowest to highest order

Hint, make use of rule #3, (log n) ^k grows slowly

First, some of the functions can be ordered through intuition:
48, N, N^2, N^3, 2^N

By comparing to these known functions, the order of the rest of the functions can be found:

$\frac{1}{N}$
Since $\frac{1}{N}$ approaches zero as N approaches infinity,

$$\frac{1}{N} < 48$$

$\sqrt{N}$
Since $\sqrt{N}$ is greater than 48 if $48^2 < N$, and $\sqrt{N}$ is less than N,

$$48 < \sqrt{N} < N$$

$N \log \log N$
Since $N < N \times a$ if a is greater than one, and $\log \log N < N^2$,

$$N < N \log \log N < N^2$$

$N \log N$
Since $\log \log N < \log N$, and $\log N < N^2$,

$$N \log \log N < N \log N < N^2$$

$N \log(N^2)$
Since $\log N < \log(N^2)$, and $\log(N^2) < N^2$,

$$N \log N < N \log(N^2) < N^2$$

$N\ (\log N)^2$
Since $\log(N^2) < (\log N)^2$ if $100 < N$, and $(\log N)^2 < N^2$,

$$N \log(N^2) < N(\log N)^2 < N^2$$

$N^{1.5}$
Since $(\log N)^2 < \sqrt{N}$ from Rule 3 in chapter 2 of the textbook, and $N^{0.5} < N$,

$$N\ (\log N)^2 < N^{1.5} < N^2$$

`

$N^2 \log N$
Using the same reasoning as in the case of $N \log N$,

$$N^2 < N^2 \log N < N^3$$

$2^{\frac{N}{2}}$
Since, as an exponential, it increases faster than the cubic function, but $\frac{N}{2} < N$,

$$N^3 < 2^{\frac{N}{2}} < 2^N$$

As a result, this order is found:
$\frac{1}{N}$
48
$\sqrt{N}$
N
$N \log \log N$
$N \log N$, which is equivalent to $N \log N$
$N \log(N^2)$
$N\ (\log N)^2$
$N^{1.5}$
N^2
$N^2 \log N$
N^3
$2^{\frac{N}{2}}$
2^N

`

Problem 5

Determine whether $f(N) = O(g(N))$, $g(N) = O(f(N))$, or both.

$$f(N) = \frac{N^2 - N + 3}{3}, g(N) = 6N$$

Initially, $f(N)$ is greater than $g(N)$, but is later exceeded by $g(N)$, and is again exceeded by $f(N)$. To find the N values where these occurs:

$$f(N) = g(N)$$

$$\frac{N^2 - N + 3}{3} = 6N$$

$$N^2 - N + 3 = 18N$$

$$N^2 - 19N + 3 = 0$$

Using the quadratic formula, the solutions are: $\frac{19-\sqrt{349}}{2}$, or about 0.159, and $\frac{19+\sqrt{349}}{2}$, or about 18.841. Therefore:

If $N < \frac{19-\sqrt{349}}{2}$ or $N > \frac{19+\sqrt{349}}{2}$, $g(N) = O\big(f(N)\big)$

If $\frac{19-\sqrt{349}}{2} < N < \frac{19+\sqrt{349}}{2}$, $f(N) = O\big(g(N)\big)$

$$f(N) = N + 2\sqrt{N}, g(N) = N^2$$

$f(N)$ is initially greater than $g(N)$, and is later exceeded. To find the N values where these occurs:

$$f(N) = g(N)$$

$$N + 2\sqrt{N} = N^2$$

$$N^{\frac{1}{2}} + 2 = \frac{N^2}{N^{\frac{1}{2}}}$$

$$N^{\frac{1}{2}} + 2 = N^{\frac{3}{2}}$$

$$2 = N^{\frac{3}{2}} - N^{\frac{1}{2}}$$

$$2 = N^{\frac{3}{2}} - N^{\frac{1}{2}}$$

`

$$0 = N^{\frac{3}{2}} - N^{\frac{1}{2}} - 2$$

Graphing software shows that $N \approx 2.315$. For this problem, it will be assumed that $N = 2.315$. Therefore:

If $N < 2.315$, $g(N) = O(f(N))$

If $N > 2.315$, $f(N) = O(g(N))$

$$f(N) = N\log N\,, g(N) = \frac{N\sqrt{N}}{2}$$

g(N) is always above f(N). Therefore:

$f(N) = O(g(N))$

$$f(N) = 2(\log N\,)^2\,, g(N) = \log N + 1$$

f(N) and g(N) alternate, with f(N) originally above, then g(N), and finally with f(N). Finding the points where they alternate:

$$f(N) = g(N)$$

$$2(\log N\,)^2 = \log N + 1$$

$$2(\log N\,)^2 - \log N - 1 = 0$$

$$2a^2 - \mathrm{a} - 1 = 0, a = \log N$$

$$(-2a - 1)(-a + 1) = 0, A = \log N$$

$$a = -\frac{1}{2}, a = 1$$

$$\log N = -\frac{1}{2}, \log N = 1$$

$$N = 10^{-\frac{1}{2}}, N = 10^1$$

The two functions alternate at $10^{-\frac{1}{2}}$, or about 0.316, and 10. Therefore,

If $N < 10^{-\frac{1}{2}}$ or $N > 10$, $f(N) = O(g(N))$

If $10^{-\frac{1}{2}} < N < 10$, $g(N) = O(g(N))$

`

$$f(N) = 4N \log N + N, g(N) = \frac{N^2 - N}{2}$$

Considering the higher order values, $g(N)$ will be higher than $f(N)$. As a result:

$f(N) = O(g(N))$

`

Problem 7

Give an analysis of the program fragments.

Fragment 1

This case is very simple, and should result in a Big-Oh of N.

N=10,000, approx. 100 microseconds

N=50,000, approx. 550 microseconds

N=100,000, approx. 1,000 microseconds

The runtime follows the N value in a predictable way, so I believe the Big-Oh of fragment 1 is N

Fragment 2

This is also straightforward, and the Big-Oh should be N^2

N=100, approx. 90 microseconds

N=200, approx. 440 microseconds

N=300, approx. 1000 microseconds

N=400, approx. 1700 microseconds

The number of microseconds is roughly equal to N/100, squared, and times 100. For example, N=400, divided by 100 is 4, squared is 16, and times 100 is 1600, which is close to the 1700 microseconds recorded.

Fragment 3

The inner loop, from the N * N, appears to produce a Big-Oh of N^2. It is enclosed by a loop that has N, so the result should be N^3.

N=10, approx. 12 microseconds

N=20, approx. 80 microseconds

N=30, approx. 400 microseconds

N=40, approx. 800 microseconds

N=50, approx. 1500 microseconds

The time roughly equals N/10, cubed, times ten. For example, N=50, divided by ten and cubed is 125, and times 10 is 1250. It is somewhat close to 1500 microseconds recorded.

`

Fragment 4

The inner loop appears similar to logN since it goes to i, which is passed in. It is enclosed in a loop that has a Big-Oh of N. Therefore, the whole algorithm should have a Big-Oh of N(logN).

N=100, approx. 80 microseconds

N=200, approx. 260 microseconds

N=300, approx. 550 microseconds

N=400, approx. 850 microseconds

N=500, approx. 1400 microseconds

The resulting time is somewhat close to N logN, if N is first divided by ten, and the result of the equation is multiplied by ten. For example, 500 is divided by ten, then applied to the function. 50*log50 is about 85, times ten is 850. Is off by a bit, but the other N values are closer.

Fragment 5

This one was complicated, but we assumed it was N log((logN)^2). The i*i, like Fragment 3, was assumed to mean a square. And since it's passed in to the inner for loop similar to Fragment 4, it's assumed to mean a logarithm. The inner most for loop goes to j, and since its format is also similar to Fragment 4, it's assumed to be a logarithm too.

N=100, approx. 5 milliseconds

N=100, approx. 9 milliseconds

N=100, approx. 19 milliseconds

N=100, approx. 35 milliseconds

`