Homework 2

1. Find an exact closed-form formula for M (n), the worst-case number of *'s performed by the algorithm below on input n:

```
int p1(int n)
{
  if (n == 0)
    return 1*2*3;

  int ans = 1;
  for (int i = 0; i < 2*n; ++i)
    ans = ans * i;
  return ans * p1(n-1);
}
```

A1: First we will translate the code to a recurrence. The base case M(0) will have a total of 2 multiplication operations ((1 * 2) * 3). For the recursive case M(n), you have a for loop that will do a multiplication for each comparison iteration and another multiplication statement within the for loop itself. You will be doing this from i = 0 to i < 2n, so 2n - 1 - 0 + 1 = 2n terms. For a for loop, we do a summation. We also have to pay for the final check in the for loop, which is one more multiplication operation. Then you will have one multiplication operation in the return statement followed by an ambiguous function call p1(n - 1), which we know the name for.

So far, you have

M(0) = 2

$M(n) = ((\sum_{i=0}^{2n-1} (2)) + 1) + (1) + M(n - 1) \rightarrow$

$M(n) = (2((2n - 1) - (0) + 1) + 1) + (1) + M(n - 1) \rightarrow$

$M(n) = (2(2n) + 1) + (1) + M(n - 1) \rightarrow$

M(n) = 4n + 2 + M(n - 1)

Now, we solve the recurrence relation.

M(0) = 2

M(1) = 4(1) + 2 + M(0) = 4 + 2 + 2 = 8

M(2) = 4(2) + 2 + M(1) = 8 + 2 + 8 = 18

M(3) = 4(3) + 2 + M(2) = 12 + 2 + 18 = 32

M(4) = 4(4) + 2 + M(3) = 16 + 2 + 32 = 50

My guess is that M(n) = 2(n + 1)^2, for n ≥ 0.

Let's check if the recurrence = guess.

For M(0): 2 =? 2(0 + 1)^2 → 2 =? 2(1)^2 → 2 =? 2 True

For M(1): 4 + 2 + 2 =? 2(1 + 1)^2 → 8 =? 2(2)^2 → 8 =? 8 True

For M(2): 8 + 2 + 8 =? 2(2 + 1)^2 → 18 =? 2(3)^2 → 18 =? 18 True

As you can see, the formula seems to hold up.

2. Find an exact closed-form formula for the number of *'s performed by the algorithm below on input n, which is a nonnegative power of 3:
   `int p2(int n)`

```
{
  if (n == 1)
    return 1*2*3;

  int ans = 1;
  for (int i = 0; i < 2*n; ++i)
    ans = ans * i;
  return ans * p2(n/3) * p2(n/3);
}
```

A2: First, translate the code into a recurrence.

M(1) = 2

M(n) = (($\sum_{i=0}^{2n-1}$ (2)) + 1) + (2) + M(n/3) + M(n/3) →

M(n) = (2((2n - 1) - (0) + 1) + 1) + (2) + M(n/3) + M(n/3) →

M(n) = 4n + 3 + 2M(n/3)

Then we solve the recurrence.

M(1) = 2

M(3) = 4(3) + 3 + 2M((3)/3) = 12 + 3 + 2(2) = 19

M(9) = 4(9) + 3 + 2M((9)/3) = 36 + 3 + 2(19) = 77

M(27) = 4(27) + 3 + 2M((27)/3) = 108 + 3 + 2(77) = 265

Using the unrolling method, do

M(n) = 1 * 4n + 1 * 3 + 2M(n/3)

= 4n + 3 + 2[4n + 3 + 2M(n/9)]

= 4n + 3 + 8n + 6 + (2^2)M(n/9) = 12n + 9 + (2^2)M(n/9)

= (3 * 4)n + (3 * 3) + (2^2)M(n/9)

= 12n + 9 + (2^2)[4n + 3 + 2M(n/27)]

= 12n + 9 + 16n + 12 + (2^3)M(n/27) = 28n + 21 + (2^3)M(n/27)

= (7 * 4)n + (7 * 3) + (2^3)M(n/27)

= 4n + 8n + 16n + 3 + 6 + 12 + (2^3)M(n/27)

= (2^2)n + (2^3)n + (2^4)n + (2^0)(3) + (2^1)(3) + (2^2)(3) + (2^3)M(n/27)

= (2^0)(3) + (2^1)(3) + (2^2)(3) + (2^2)n + (2^3)n + (2^4)n + (2^3)M(n/27)

= 3[(2^0) + (2^1) + (2^2)+..+(2^n)] + n[(2^2) + (2^3) + (2^4)+..+(2^{n+2})] + (2^3)M(n/27)

= 3[((2^{n+1} - 1) / (2 - 1))] + n[(2^0 + 2^1 + (2^2) + (2^3) + (2^4)+..+(2^n)) + (2^{n+1}) + (2^{n+2}) - 2^0 - 2^1] + (2^3)M(n/27)

= 3[((2^{n+1} - 1) / (2 - 1))] + n[((2^{n+1} - 1) / (2 - 1)) + (2^{n+1}) + (2^{n+2}) - 2^0 - 2^1] + (2^3)M(n/27)

= 3[((2^{n+1} - 1) / (2 - 1))] + n[((2^{n+1} - 1)) + (2^{n+1}) + (2^{n+2}) - 3] + (2^3)M(n/27)

Last part must cancel out the function call.

= 3[((2^{n+1} - 1) / (2 - 1))] + n[((2^{n+1} - 1)) + (2^{n+1}) + (2^{n+2}) - 3] + (2^floor(cuberoot(n)))M(n/n)

= 3[((2^{n+1} - 1) / (2 - 1))] + n[((2^{n+1} - 1)) + (2^{n+1}) + (2^{n+2}) - 3] + (2^floor(cuberoot(n)))M(1)

= 3[((2^{n+1} - 1) / (2 - 1))] + n[((2^{n+1} - 1)) + (2^{n+1}) + (2^{n+2}) - 3] + (2^floor(cuberoot(n))) * (2)

M (n) = 3[((2^{n+1} - 1) / (2 - 1))] + n[((2^{n+1} - 1)) + (2^{n+1}) + (2^{n+2}) - 3] + (2^floor(cuberoot(n)) + 1)

3. Find the asymptotic growth rate of M (n), which is defined by the following recurrence:

M (1) = 1

M (n) = 3 M (n / 4) + f (n) , n = 4 , 4^2 , 4^3 , …

when

1. f (n) = 2020 ;
2. f (n) = square root(n) + $n^{\log_4 3}$ – 5 $n^{3/4}$;
3. f (n) = n + lg n + $n^{\log_4 3}$

A3: We will be using the Master Theorem to help us out

1) M (n) = 3 M (n / 4) + 2020 , n = 4 , 4^2 , 4^3 , …

a = 3 ≥ 1, b = 4 ≥ 2, c = 1 ≥ 0, d = 0. Since 3 ? 4^0 → 3 > 1 → a > b^d, and so that is case 2 of the Master Theorem, which means that M(n) ∈ $\Theta(n^{\log_b a})$ → M(n) ∈ $\Theta(n^{\log_4 3})$.

2) M (n) = 3 M (n / 4) + (n)^(½) + $n^{\log_4 3}$ – 5 $n^{3/4}$, n = 4 , 4^2 , 4^3 , …

a = 3, b = 4, c = 1, d = (3 / 4). 3 ? 4^($\log_4 3$) → 3 = 3 → a = b^d, and so that means that M(n) ∈ Θ(n^d(lg n)).

3) M (n) = 3 M (n / 4) + n + lg n + $n^{\log_4 3}$, n = 4 , 4^2 , 4^3 , …

a = 3, b = 4, c = 1, d = (log 10). 3 ? 4^(log 10) → 3 < 4 → a < b^d, and so that means that M(n) ∈ Θ(n^d).