Math 135: Sample Midterm 2

1. (10 points) Let $f(n) = 2n \log_2(2n)$ and let $g(n) = n \log_2 n^2$. Prove that $f(n) = \Theta(g(n))$.

2. (10 points) Find a recurrence for $f(n)$, the number of bitstrings of length n that do not have three consecutive ones. For example, $f(3) = 7$ because out of the 8 bitstrings of length 3, only one has three consecutive ones.

   $$\{000, 001, 010, 011, 100, 101, 110, \cancel{111}\}$$

   You do <u>not</u> have to solve the recurrence - just give the recurrence and explain your answer.

3. (5 points per part, 10 points total) Give the general form for the following recurrences. Note that you do <u>not</u> have to solve for constants. For example, the general form for the recurrence $R(n) = 3R(n-1) + 5$ would be $R(n) = c_1 3^n + c_2$, where c_1 and c_2 are constants.

   (a) $A(n) = (-3)A(n-1) + 4 \cdot 5^n$

   (b) $B(n) = B(n-1) + 2B(n-2) + 2^n(n-2)$

4. (10 points) Let $T(n) = 6T(\frac{n}{6}) + n^2$, with $T(1) = 1$. Solve the recurrence $T(n)$ <u>asymptotically</u>. Show your work. You may use any theorems or methods presented in class. If you use the guess and check method, then you must prove that your answer is correct. If you use other methods from class, then you do not need to prove your answer is correct.

5. (15 points) Let $T(n) = \sqrt{n}T(\sqrt{n}) + n$, with $T(2) = 1$. Solve the recurrence $T(n)$ <u>asymptotically</u>. Show your work. You may use any theorems or methods presented in class. If you use the guess and check method, then you must prove that your answer is correct. If you use other methods from class, then you do not need to prove your answer is correct.

6. Analyze the runtime of the following algorithms. ("Analyze the runtime" means give a simple function $f(n)$ such that the runtime of the algorithm is $\Theta(f(n))$.) You should explain how you got your answers, but you do not need to prove that your answers are correct.

   (a)

   ```
   ALGA(n):
     if n = 0 or n = 1 then
         return 1
     else
         x ← ALGA(n − 1)
         y ← ALGA(n − 2)
         return 2x + 3y
   ```

   (b)

   ```
   ALGB(A[1...n]):
     i ← ⌈n/3⌉ + 1
     j ← ⌊2n/3⌋
     if n = 1 then
         return A[1]
     else
         x ← ALGB(A[1...j])
         y ← ALGB(A[i...n])
         z ← 0
         for k = i to j do
             z ← z + A[k]
         return xy + z
   ```

(c)

```
ALGC(n):
  x ← n
  i ← 2
  while i < n do
      x ← x + 3i − 5
      i ← i^2
  return x
```