The exam will be open-books and open-notes and will have very similar problems to this, but it will be shorter (about 10 problems). The real exam will not contain nearly as many hints as this sample exam. *Be careful! The real exam may contain some of these exact problems, but it may also contain problems that are very similar but not identical to these problems, and it will certainly contain problems that are not exactly represented here!*

1. Consider the statement "x^3 is an upper bound on $4x^2 + 3x$, but it is not a tight bound."
   (a) Rewrite the statement using the most appropriate mathematical notation. (Hint: read the statement very carefully.)
   (b) Give a complete formal proof of the statement. (Hint: Use the limit theorem.)
2. Repeat the previous problem with the statement "x^3 is a tight bound on $4x^3 + 3x^2 - 45x + 19$."
3. Consider the following segment of code.

```
if(!( x<=0 || x>0 ) ) {
   doSomething();
} else {
   doAnotherThing();
}
```

   (a) Write the condition in the if statement as a logical expression, defining predicate(s) as necessary. (Hint: You should only need to define one predicate.)
   (b) Rewrite the code so that it is as simple as possible and logically equivalent. *Explain why it is equivalent.*
4. Consider the following code. Regard the input size as n (so $n \geq 0$) and assume that `power(a,i)` takes i operations to compute a^i.

```
double addPowers(double a, int n) {
    if(a==1) {
        return n;
    } else {
        double sum = 0;
        for(int i=0;i<n;i++) {
            sum += power(a,i);
        }
        return sum;
    }
}
```

   (a) Express what `addPowers(a,n)` computes when $a \neq 1$ using the appropriate mathematical notation.
   (b) What is the worst-case complexity of `addPowers`? (Hint: Don't confuse what the output of the algorithm is with the complexity of the algorithm. The previous question is asking what the algorithm computes. This question is asking how long it takes to compute it. You might get confused because there are summations involved in both (bonus hint for part (a)!), but the summations are completely different.)
   (c) Rewrite the code so it is as efficient as possible.
   (d) Give the worst-case complexity of your version of the algorithm. (Hint: It should be faster than the original!)
5. Find and prove a *simple tight bound* on the recurrence $T(n) = 10T(n/3) + 8n^2, T(1) = 1$. (Hint: This problem should be very easy! If you spend more than a few minutes on it then you are doing it the hard way.)
6. Find and prove an *exact formula* for the recurrence $T(n) = 2T(n-1) - 1$, $T(1) = 2$. (Hint: Use the iteration method. It will involve some algebra.)
7. Express the following phrase using quantifiers. *Your solution should contain no English words.*

   "There are constants c and x_0 such that for all $x \geq x_0$, $f(x)$ is no greater than $c \cdot g(x)$."

8. Use induction to prove that for all $n \geq 2$, $\displaystyle\sum_{k=2}^{n} \frac{1}{(k-1)k} = \frac{n-1}{n}$.

9. Let $B(n)$ be the worst-case running time of `Blah(n)`. Give a recurrence relation for $B(n)$. ***Do not solve the recurrence relation.***

```
int Blah(int n) {
    if(n>0) {
        int c = Blah(n-1);
        for(int i=0;i<n;i++) {
             c++;
        }
        c += Blah(n-1);
        return c;
   } else {
      return 0;
   }
}
```

10. Algorithm A has a complexity of $\Theta(n^2)$ and algorithm B has a complexity of $\Theta(n \log n)$.

    (a) Which algorithm is more efficient? Explain.

    (b) Is it *always* more efficient? Explain.

11. Let A be the set of all women, and B be the set of all disc golfers. What are each of the following: $A \cup B$, $A \cap B$, $A \setminus B$, $B \setminus A$, $\overline{A}$?

12. Compute the following sums without just adding all of the terms. In other words, use some of what you learned this semester. $\displaystyle\sum_{k=11}^{40} k^2$, $\displaystyle\sum_{k=20}^{80} (k^3 - 10)$, $\displaystyle\sum_{k=5}^{25} k(k+2)$

13. Let G be a connected graph with n vertices.

    (a) What is the minimum number of edges G can have?

    (b) What is the maximum number of edges G can have?

14. Describe but an adjacency matrix for K_n would look like.

15. Give two reasons that a graph with degree sequence of 8, 6, 4, 3, 3, 2, 2, 1, cannot exist.

16. Let G be a graph on n vertices. An **adjacency matrix** M for a graph can be treated like a 2-dimensional array, so if $M[u][v] = 1$, (u, v) is an edge and if $M[u][v] = 0$, (u, v) is not an edge. Similarly, if L is an **adjacency list** representation of M, then for a vertex u, $L[u]$ is a list of vertices that are connected to u.

    (a) Give an algorithm to determine whether or not (u, v) is an edge of G (where u and v are vertices), assuming G is stored using an **adjacency matrix** M. *Give the complexity of your algorithm.*

    (b) Give an algorithm to determine deg(u) (the degree of vertex u), assuming G is stored using an **adjacency matrix** M. *Give the complexity of your algorithm.*

    (c) Repeat both (a) and (b) assuming G is stored using an **adjacency list**.