Answer all questions. Give informal (at least) proofs for all answers. Grading will be on completeness and logical correctness, and if applicable, efficiency, as well as correctness. Out of 80 points.

Order questions- 10 points each For each, answer True or False, and give a short explanation for your answer.

1. $n^2 \in O(2^n)$.  
   Yes, $n^2 \leq 2^n = 1 \times 2^n$ for $n \geq 2$, so $n^2 \in O(n2^n)$.
2. $n^2 \in \Theta(2^n)$.  
   False. To be $\Theta$, you need to be of the same order, not just bounded above. Since $n^2/2^n$ goes to 0 as $n$ increases, $n^2 \in O(2^n)$, not $\Theta(2^n)$.

3. If $f$ and $g$ are functions from positive integers to positive integers, and $f(n) \in O(g(n))$, then $f(n) + g(n) \in O(g(n))$.  
   True. If $f(n) \in O(g(n))$, then $f(n) \leq cg(n)$ for $n \geq N$ for some constants $c, N > 0$. Then $f(n) + g(n) \leq cg(n) + g(n) = (c + 1)g(n)$ for $n \geq N$. So we can use the constant $c' = c + 1$ in the definition of $O$ to see $f(n) + g(n) \in O(g(n))$.

4. If $f$ and $g$ are functions from positive integers to positive integers, and $f(n) \in O(g(n))$, then $2^{f(n)} \in O(2^{g(n)})$.  
   False. If $f(n) = 2n$ and $g(n) = n$ then $f(n) \in O(g(n))$, but $2^{f(n)}/2^{g(n)} = 2^{2n}/2^n = 2^n$ which goes to infinity as $n$ increases, so $2^{f(n)}$ is not in $O(2^{g(n)})$.

Analyzing loops-20pts Here is an algorithm that, given two sorted lists $A[1..n], B[1..n]$, decides whether the two sets of values intersect, i.e., are there any $1 \leq i, j \leq n$ so that $A[i] = B[j]$?

Intersect $(A[1..n], B[1..n]$: sorted lists of positive integers)

1. $I \leftarrow 1$, $J \leftarrow 1$, $Found \leftarrow False$.
2. While $I \leq n$ AND $J \leq n$ and $Found = False$ do:
3. Return $Found$.

Give a time analysis, up to order, for this algorithm. Be sure to explain your answer.

Each iteration of the while loop, either $I$ or $J$ is incremented, or we set $Found$ to true and exit the loop. Thus, if we iterate more than $2n$ times, either $I$ or $J$ will be greater than $n$, at which point we exit the loop. Thus, there are $O(n)$ iterations. Since all operations within the loop and initialization are constant time, the total time is thus $O(n)$.

Correctness Proof: 20 points Assume there is a pair of indices $i, j$ so that $A[i] = B[j]$. Prove the following loop invariant for the algorithm above: at all times, $I \leq i$ and $J \leq j$. (Hint break it up into cases based on whether, before the loop, $I = i$ or $I < i$ and whether $J = j$ or $J < j$.

At the start of the algorithm, $I = J = 1$. Since we are told $1 \leq i, j \leq n$, $I \leq i$ and $J \leq j$ at the start. Assume at the beginning of an iteration, $I \leq i$ and $J \leq j$. Then if neither is incremented in the loop, this is still true at the end.

If $I$ is incremented, then $A[I] < B[J]$. Since $J \leq j$ and $B$ is sorted, $B[J] \leq B[j] = A[i]$. Thus, $A[I] < A[i]$, so $I < i$, and after we increment, we will still have $I \leq i$ (since both are integers.)

If $J$ is incremented, then $A[I] > B[J]$. Since $I \leq i$ and $A$ is sorted, $A[I]J \leq A[i] = B[j]$. Thus, $B[J] < B[j]$, so $J < j$, and after we increment, we will still have $J \leq j$ (since both are integers.)
In all cases, the invariant $I \leq i$ and $J \leq j$ remains true.

Although its not necessary for the quiz, I’ll just quickly show how the invariant proves the algorithm correctly determines whether the lists intersect. If the algorithm returns True, then there must be an iteration where $A[I] = B[J]$, since that is the only case where $Found$ is set to true. In this case, $A[I] = B[J]$ is in the intersection. If there is an element in the intersection, we have some $A[i] = B[j]$. Then by the loop invariant, at all times $I \leq i \leq n$ and $J \leq j \leq n$. Thus, the only way the algorithm terminates is if $Found$ is True, and in this case, the algorithm returns True. Thus, the algorithm returns True if and only if the lists intersect.