1. (a) We show that $\overline{\text{UpToOneSAT}} \in NP$; the result follows.

   $$\overline{\text{UpToOneSAT}} = \{\varphi \in CNF | \text{there exist} \geq 2 \text{ satisfying truth assignments to } \varphi\}$$

   Consider an instance φ. We guess two truth assignments and check that they are different and both satisfy φ.

   (b) We show that $\text{SAT} \leq_L \overline{\text{UpToOneSAT}}$; the result follows.
   Consider an instance φ of SAT and let $\varphi' = \varphi \wedge (z \vee \neg z)$ where z is a new variable (obviously a log-space reduction).

   ($\Rightarrow$) Let ρ be a satisfying truth assignment for φ. Then both ρ_{true} and ρ_{false} satisfy φ' where $\rho_v(x) = \rho(x)$ for $x \neq z$ and $\rho_v(z) = v$.

   ($\Leftarrow$) φ is a sub-formula of φ', hence any truth assignment satisfying φ' satisfies φ as well.

2. A d-regular graph cannot contain a clique of size $d+2$, since the latter is a $(d+1)$-regular graph. Here's a polynomial algorithm that solves the problem:

   Given an instance (G, k), we answer $false$ if $k > d+1$ or $n = |V| < k$. Otherwise, we check the $\binom{n}{k} \leq n^k \leq n^{d+1}$ sets of size k in G and answer $true$ if any of them is a clique.

   Time Complexity: $\leq \binom{k}{2} n^{d+1} = O(n^{d+1})$.

   The result follows for $d = 2005$.

   Note: Can you find a $\binom{d}{k}\binom{k}{2} n = O(n)$ algorithm?

3. Let $G = (A, B, E)$ be a bipartite graph where $A = \{a_i\}_{i=1}^r$, $B = \bigcup_{i=2}^r B_i$, $B_i = \{b_j^i | ij \leq r\}$, $E = \{(a_{t+(i-1)j}, b_j^i) | 1 \leq t \leq j, 2 \leq i \leq r, ij \leq r\}$. Every vertex $b_j^i \in B_i$ has degree i; every vertex $a_k \in A$ has degree $< r$ since it has at most one neighbour from B_i for each $2 \leq i \leq r$.

   The greedy algorithm chooses first all vertices of B_r (of maximal degree r) to the vertex cover. Eliminating these edges from the graph, the degree of every vertex in A is $\leq r-2$. The greedy algorithm chooses now all vertices of B_{r-1} (of maximal degree $r-1$) to the vertex cover. This continues until the graph has no edges, and the algorithm returns the vertex cover B of size $|B| = \sum_{i=2}^r \lfloor \frac{r}{i} \rfloor = \Theta(r \log r)$, where the optimal vertex cover is A of size r. The approximation ratio of the algorithm, therefore, is $\Omega(\log r)$.

   The result follows for $r = 16$ and $|B| = 34 > 2r$.

4. Similar to what we do when composing log-space reductions, we simulate M_f's run on $g(x)$ and compute each bit of $g(x)$ when M_f tries to access it by the following method:
   Simulate M_g on x. Whenever M_g writes a bit, advance a counter; when the requested bit arrives, remember it. Continue running until either M_g accepts (and then return the requested bit of $g(x)$ to M_f) or rejects (and then reject as well).

   Correctness:
   We never return a wrong bit of $g(x)$ and there's always an execution path that of M_g that returns the correct bit. Therefore, there's always an execution path of M_f that returns the correct answer.

   Space Complexity:
   Simulating M_g takes $O(\log |x|)$, remembering which bit of $g(x)$ is needed takes $O(\log |g(x)|)$ and simulating M_f takes $O(\log |g(x)|)$ as well. We have $|g(x)| = poly(|x|)$ since M_g uses log-space, hence $O(\log |g(x)|) = O(\log |x|)$.

5. Let $e_1, e_2, \ldots, e_m$ be the edges ordered by increasing weights (two edges with identical weights retain the original order defined by the input). The ith of the m iterations performed in Kruskal's algorithm is done as follows:

   For each edge $e_i = (u_i, v_i)$ we check whether u_i and v_i are connected[1] in the graph $G_i = (V, E_i), E_i = \{e_1, \ldots, e_i\}$. If not, write e_i to the output.

   Correctness:
   Implied by Kruskal's algorithm. Note that actually we should have checked whether u_i and v_i are connected in $MST(G_i)$, but since this is a spanning tree, the answer is the same.

   Space Complexity:
   Obviously the order of E given in the input is not necessarily $e_1, e_2, \ldots, e_m$, but we only have to remember e_i in order to determine E_i, and we can find e_1 and advance from e_i to e_{i+1} using logarithmic space as well.

6. (a) Let $G = (V, E)$ be a tournament graph without directed 3-cycles. Assume that G contains some directed cycle and let C be a cycle of minimal length in G. G is simple, has neither anti-parallel edges nor directed 3-cycles, so $|C| \geq 4$. Thus, $C = u \to v \to x \to y \rightsquigarrow u$. The edge $(x, u) \notin E$ since G has no directed 3-cycles; hence $(u, x) \in E$ and $C' = u \to x \to y \rightsquigarrow u$ is a shorter cycle than C, contradicting our choice of C.

   (b) Let $G = (V, E)$. We examine all triplets $u, v, w \in V$; if $u \to v \to w \to u$ in G we add u, v, w to our vertex cover and delete them from the graph.

   Correctness:
   After the algorithm is done, the graph remains directed 3-cycle free, hence by 6a directed cycle free.

   Time Complexity: $O(|V|^3)$, polynomial.

   Approximation Ratio:
   Our algorithm returns a vertex cover of size $3k$ when it finds k vertex-disjoint directed 3-cycles. Every algorithm solving the problem must remove at least one vertex from each such cycle, so it returns at least k vertices. Therefore, the approximation ratio is ≤ 3.

[1] run both s-t-Con and $\overline{\text{s-t-Con}}$. if both return $false$, reject; otherwise return the (100% correct) answer.