1. (a) Assume $A, B \in NL$, i.e., there exist NDTMs M_A, M_B deciding A, B respectively in logarithmic space. We describe a NDTM deciding $A \cup B$ in logarithmic space:

Algorithm 1 DECIDE-A-OR-B(x)

guess $m \in \{0, 1\}$
if $m = 0$ **then**
  return $M_A(x)$
else
  return $M_B(x)$

Correctness:
($\Rightarrow$) If $x \in A \cup B$, then either $x \in A$ or $x \in B$. If $x \in A$, there's an execution path of M_A that returns *true*, so by guessing $m = 0$ and following that path, DECIDE-A-OR-B returns *true*. The case $x \in B$ is analogous.
($\Leftarrow$) If DECIDE-A-OR-B returned *true*, it has found an execution path in either M_A or M_B that returns *true*, hence $x \in A$ or $x \in B$. In any case, $x \in A \cup B$.
Space Complexity: Only constant space more than $M_A, M_B \Rightarrow$ logarithmic space.

(b) Using $NL =$ co-NL and Question 1a,

$$A, B \in NL \Rightarrow \overline{A}, \overline{B} \in NL \Rightarrow \overline{A} \cup \overline{B} \in NL \Rightarrow A \cap B = \overline{\overline{A} \cup \overline{B}} \in NL$$

2. (a) SAME-SCC$(G, u, v) =$ (S,T)-CON$(G, u, v) \wedge$ (S,T)-CON(G, v, u). Since (S,T)-CON $\in NL$, a reasoning similar to the one used in Question 1 leads to SAME-SCC $\in NL$.

(b) $NL =$ co-NL, hence DIFFERENT-SCC $= \overline{\text{SAME-SCC}} \in$ co-NL. For some fixed k, let $L_{\geq k} = \{\langle G \rangle |$ the directed graph G contains $\geq k$ SCCs$\}$. Consider the following algorithm:

Algorithm 2 AT-LEAST-k-SCCS(G)

for $i = 1$ to k **do**
  guess $v_i \in V$
  for $j = 1$ to $i - 1$ **do**
    if DIFFERENT-SCC$(v_i, v_j) = false$ **then**
      return $false$
return $true$

Correctness:
($\Rightarrow$) If G has $< k$ SCCs, then for every choice of $v_1, \ldots, v_k$, there exist $j < i$ such that v_i, v_j are in the same SCC, and for that pair, every execution path of DIFFERENT-SCC returns $false$. Thus, all execution paths return $false$.
($\Leftarrow$) If G has $\geq k$ SCCs, then when choosing $v_1, \ldots, v_k$ from distinct SCCs, for all $1 \leq j < i \leq k$ some execution path of DIFFERENT-SCC(v_i, v_j) returns *true*.Thus, there exists an execution path that returns *true*.
Space Complexity:
AT-LEAST-k-SCCS has to remember k vertices ($k \log n$) and DIFFERENT-SCC uses logarithmic space $\Rightarrow$ logarithmic space.

AT-LEAST-k-SCCS accepts $L_{\geq k}$ in logarithmic space, so $L_{\geq k} \in NL$ for all $k \in \mathbb{N}$, esp. for $k = 2006, L_{\geq 2006} \in NL$.

(c) Using $NL = \text{co-}NL$, Question 1b and Question 2b,

$$L_{=k} = L_{\geq k} \cap L_{<(k+1)} = L_{\geq k} \cap \overline{L_{\geq (k+1)}} \in NL$$

3. Let $L_2 = \{\langle G \rangle | \ G \text{ is bipartite}\}$. G is bipartite if and only if G contains no odd cycles, therefore $\overline{L_2} = \{\langle G \rangle | \ G \text{ contains an odd cycle}\}$. A simple modification of Question 1 from Exercise 2 shows $\overline{L_2} \in NL$, hence $L_2 \in \text{co-}NL = NL$.

4. Let $A \in NP$ and let M_A be a NDTM deciding A in polynomial time. Fix an input x. M_A runs $\leq poly(|x|)$ steps, so at most $poly(|x|)$ non-deterministic decisions are made throughout any execution path. Each decision has a constant number of alternatives ($\leq 2|Q||\Gamma|$), so encoding all decisions taken during some (specific) execution path as y takes $poly(|x|)$ space.

   Given (x, y) as input, the DTM M will simulate $M_A(x)$ where decisions are made based on y's contents (if y ends prematurely then M returns $false$). M accepts $x \Leftrightarrow$ for some $y, |y| = poly(|x|)$ $M(x, y)$ returns $true \Leftrightarrow$ some execution path of $M_A(x)$, encoded by y, returns $true$. We've shown $A \in NP^*$, hence $NP \subseteq NP^*$.

5. Let $A \in P$. Since $CVAL$ is P-complete, $A \leq_L CVAL$ using some logarithmic reduction f. Assuming $CVAL \in L$, we can simulate the TM solving $CVAL$ on $f(x)$ without precomputing $f(x)$ (same process used in logarithmic reductions chaining), hence deciding A in logarithmic space. Thus $P \subseteq L$. Together with the known result $L \subseteq P$, we get $L = P$.

6. - $|\{f : X \to \{0,1\}\}| = 2^{|X|}$; n input bits $\Leftrightarrow 2^n$ input values $\Rightarrow |X| = 2^n$. Thus, the number of boolean functions on n bits is 2^{2^n}.
   - For each of the $m = c\frac{2^n}{n}$ gates we have to choose its type (3 options), its first input ($\leq m + n$ options) and its second (if exists) input ($\leq m + n$ options). Therefore, the number of circuits with m gates and n inputs is at most $3^m(m+n)^{2m}$.
   - We presume $c < 1$ and compare the log of the two numbers: $\log_2 2^{2^n} = 2^n$;

$$\begin{aligned}
\log_2\left(3^m(m+n)^{2m}\right) &= m\log_2 3 + 2m\log_2(m+n) \\
&< 2m + 2m(\log_2 m + \log_2 n) \\
&< 2c2^n + 2c\frac{2^n}{n}(\log_2 c + n - \log_2 n) + 2c\log_2 n \\
&< 2c(2^n + 2^n + \log_2 n) < 5c \cdot 2^n
\end{aligned}$$

   For $c \leq \frac{1}{5}$ we can see that there're more possible boolean functions on n bits than circuits of size $\frac{2^n}{5n}$, hence some function cannot be computed by a circuit of size $\frac{2^n}{5n}$.