Name ______________________ Student No. ______________________

No aids allowed. Answer all questions on test paper. Use backs of sheets for scratch work.

Each question worth 10, and the Bonus Question worth 5

1. Recall the definition of **NP** languages in terms of a polynomial time verifier V: $L \in \mathbf{NP}$ iff there exists a polynomial time (in $|x|$) verifier $V(x, y)$ such that $L = \{x|\exists y V(x, y)\}$.

   (a) The y in the above definition is called a *witness*. Show that if $L \in \mathrm{NP}$, then there exists a fixed polynomial $p(n)$ such that if $x \in L$, then x has a witness y such that $|y| \leq p(|x|)$. In other words, if a witness exists, it is short.

   (b) Show that if $\mathbf{P} = \mathbf{NP}$ then the witness y can in fact be computed in polynomial time (in $|x|$).

   Solution: (a) This follows directly from the definition of the verifier $V(x, y)$ which has to run in time polynomial in $|x|$; therefore, regardless of what y is, V will only consider its initial segment of length at most polynomial in $|x|$. Therefore, we might as well just use that initial segment as the witness.

   (b) Suppose $L = \{x|\exists y V(x, y)\}$, where V is a polytime verifier. By (a) we know that $|y| \leq p(|x|)$, for some fixed polynomial p. Therefore, $L' = \{\langle x, w\rangle|\exists y V(x, wy)\} \in \mathbf{NP}$, since it has a polytime verifier $V'(\langle x, w\rangle, y) = V(x, wy)$. If $\mathbf{P} = \mathbf{NP}$, then $L' \in \mathbf{P}$, so for any $\langle x, w\rangle$ we can determine in polytime in $|x|$ whether it is in L'. So given x, we compute its witness (with respect to L) as follows: first determine whether $x \in L$. If no, stop. If yes, ask if $\langle w, 0\rangle \in L'$. If yes, we know that the first bit of y is 0. If no, we know that the first bit of y is 1. We continue this way, now querying $\langle x, y_1 0\rangle$, etc. We go on for $p(|x|)$ many steps.

2. A graph is *strongly connected* if every two nodes are connected by a directed path in each direction. Let:

$$\text{STRONGLY-CONNECTED} = \{\langle G \rangle |\ G \text{ is a strongly connected graph }\}$$

Show that STRONGLY-CONNECTED is **NL**-complete.

Solution: We know that PATH is **NL**-complete. So, all we have to do is show that PATH $\leq_L$ STRONGLY-CONNECTED; since composition of logspace functions is a logspace function (Theorem 8.18 in the textbook), hardness follows from this.

On the other hand, STRONGLY-CONNECTED in in **NL** because we keep two counters for enumerating all pairs of nodes, and guess a path between them, keeping track only of the last node visited on the path.

So now we show that PATH $\leq_L$ STRONGLY-CONNECTED. Take $\langle G, s, t \rangle$, an instance of PATH, and convert it to an instance of STRONGLY-CONNECTED by connecting every node (except t) to s, and connecting t to every other node (except s). Note that since these are directed edges, if there was not path from s to t, these new edges will not create a path from s to t. Correctness of the construction follows immediately from this observation.

It remains to convince ourselves that this transformation can be carried out in logspace. Assume that G is given as an adjacency matrix. Our transformation puts a 1 in every position in the t-row (except the position corresponding to the s-column), and vice-versa. Finding the t-row and s-column can be done with logspace pointers to the input.

3. Let MAX-CLIQUE $= \{\langle G, k\rangle |$ the *largest* clique of G has k vertices $\}$. (It is unknown whether MAX-CLIQUE is in **NP**.)

   (a) Show that MAX-CLIQUE is in $\mathbf{NP}^{\mathrm{SAT}}$.

   (b) Show that if $\mathbf{P} = \mathbf{NP}$, then MAX-CLIQUE is in **P**.

   Solution: (a) Both **NP** and **coNP** are in $\mathbf{NP}^{\mathrm{SAT}}$. The problem of testing whether a graph has a clique of size k in in **NP**. The problem of testing whether a graph does *not* have a clique of size $k+1$ is in **coNP**.

   (b) If $\mathbf{P} = \mathbf{NP}$, then since **P** is closed under complementation, we know that **coNP** is also equal to **P**, so by the above argument, we know that MAX-CLIQUE would be in **P**.

4. The n-input *parity function* $\text{par}_n : \{0,1\}^n \longrightarrow \{0,1\}$ outputs 1 if an odd number of 1s appear in the input variables. Show that par_n can be computed with $O(n)$ circuits.

   Solution: Assume that $n = 2^m$, because if it is not, then we can pad it with zeros (which do not change the value of the parity function) to the next closest power of 2; this doubles the size of input gates, at most.

   Now construct a balanced tree of XOR's. This tree has depth m, so size n. However, our circuits allow only AND, OR, and NOT gates. So we simulate each XOR as follows: $x \oplus y = (x \wedge \neg y) \vee (\neg x \wedge y)$. This introduces 5 gates per XOR gate, and so it increases the size of the circuit by a constant.

5. Prove that **PSPACE** = **NPSPACE**.

   Solution: Savitch's theorem says that NSPACE($f(n)$) $\subseteq$ SPACE($f^2(n)$). The result follows immediately from this.

6. BONUS QUESTION

   Use the "padding argument" to show that

   if **NEXPTIME** $\neq$ **EXPTIME** then **P** $\neq$ **NP**.

   Solution: We show the contrapositive. Assume that $\mathbf{P} = \mathbf{NP}$, and let L be any language in **NEXPTIME**. Consider $L' = \text{pad}(A, 2^{n^k})$, where 2^{n^k} is the running time of a decider for L. Then, L' can be decided in **NP** (by the same decider that ignores the padded portion of the input—but checks that the padded portion of the input is there, and that it is of the appropriate length). Thus, by assumption, $L' \in \mathbf{P}$, and now reversing the argument, it follows that L is in **EXPTIME**.