1. Answer True or false, tick the right choice or fill up as necessary (negative 2 for each incorrect answer) (3 × 10)

(a) If \( P = NP \) then \( coNP = NP \)  

\( P = NP \) then \( NP \) is closed under complementation.

(b) If \( L_1 \in P \) and \( L_2 \in NP \) then
(i) \( L_1 \leq_{\text{polytime}} L_2 \)  
(ii) \( L_2 \leq_{\text{polytime}} L_1 \)  
(iii) Neither

Note that both \( L_1, L_2 \neq \emptyset, \Sigma^* \).

Since \( L_1 \in P \), we can actually compute if \( w \in L_1 \) in polynomial time and then map \( w \) to \( x \) where \( x \in L_2 \). Similarly for \( w \not\in L_2 \). Note that \( L_2 \) is not known to be in \( P \) we we cannot do this in the other direction.

(c) The satisfiability of a boolean formula given in Disjunctive Normal Form (disjunction of minterms) is in
(i) \( \sqrt{P} \)  
(ii) In \( NP \) but not known to be in \( P \)

In DNF, we only have to pick up and satisfy a minterm. For example, if it is \( x_1 \land \bar{x}_2 \land x_3 \) then we can assign \( x_1 = True \) and \( x_2, x_3 = False \).

(d) If \( L \) is NP-complete and \( L \in DSPACE(\sqrt{n}) \), then \( NP \subset DTIME(2^{O(\sqrt{n})}) \). (Give a bound that is the best known)

(e) Let \( G_1 \) contain the productions \( S \Rightarrow AS|b \quad A \Rightarrow a|e \).
Let \( G_2 \) contain the productions \( S \Rightarrow AS|b \quad A \Rightarrow a \).

Then (i) \( L(G_2) \subset L(G_1) \) (strict subset)  
(ii) \( L(G_1) = L(G_2) \)
We can eliminate \( e \) production by having \( S \Rightarrow AS\mid S \) which is the same as \( G_1 \).

(f) \( PSPACE = NPSPACE \).  

\( PSPACE = NPSPACE \).

Follows from Savitch’s theorem that \( NSPACE(f(n)) \in DSPACE(f^2(n)) \).

(g) \( L^{rev} \) contains all the reversal of the strings in \( L \). Which of the following classes are closed under reversal
(i) Regular but not CFL  
(ii) CFLs but not recursively enumerable  
(iii) recursively enumerable
First reverse \( w \) and then run the TM for \( L \) on \( w^{rev} \).

(h) Give an example of a language \( L \) such that both \( L \) and \( \bar{L} \) (complement) are not CFLs. \( L = \{0^n|n \geq 1\} \)

(i) Given a clause \( C = (x_1 \lor x_2 \lor \bar{x}_3 \lor x_4 \lor \bar{x}_5) \) express the satisfiability of \( C \) in 3-CNF using at most two extra boolean variables.
\( (x_1 \lor x_2 \lor y_1) \land (\bar{y}_1 \lor \bar{x}_3 \lor y_2) \land (y_2 \lor x_4 \lor \bar{x}_5) \).

(j) Let \( L \) be any co-NP language. Then \( L \) is reducible to \( \overline{L_{SAT}} \) (complement of \( L_{SAT} \)) in polynomial time.  

\( L_{SAT} \subseteq co-NP \).

If \( L_1 \leq L_2 \) then \( \bar{L}_1 \leq \bar{L}_2 \) using the same reduction function by definition.
2. Let $L_1, L_2$ be two languages. Then

$$L_1/L_2 = \{ x | x \cdot y \in L_1 \text{ for some } y \in L_2 \}$$

If $L_1$ is regular and $L_2$ is CFL, then is $L_1/L_2$ regular? (10)

Since $L_1$ is regular consider a DFA $M_1$ that recognises $L_1$. Let $L = L_1/L_2$ - suppose $w \in L$. Then there exists at least some $y \in L_2$ such that $w \cdot y \in L_1$. Stated otherwise

$$w \in L \iff \exists y \ \delta(q_0, w \cdot y) \in F$$

Where $M_1 = (Q, q_0, \delta, F)$.

Suppose $\delta(q_0, w) = p$. Then for any $w'$ such that $\delta(q_0, w') = p$, $w' \in L$. We can therefore design a DFA $M$ for $L$ by using the same state diagram as $M_1$ and define the final states $F'$ as those states $q$ such that there exists some string $y \in L_2 \ \delta(q, y) \in F$.

Note: To determine which states of $M$ are in $F'$, we do not need to use any properties of $L_2$, so the above construction works for any language $L_2$, not just CFLs. That is, the above suffices as an existential proof.

However, if $L_2$ is CFL we can design an algorithm by observing that intersection of $L_2$ with regular expression (the strings that take us between two states of a DFA) is CFL. Then we can use the emptiness algorithm on the intersection. This observation was not necessary for a correct solution.

A number of solutions went astray by trying to construct such a machine using hitherto unproven claims about CFL’s having regular expressions, complicated state-transition diagram mixing NFA with reversed CFLs that was completely unnecessary and also incorrectly argued. These were penalized accordingly.
3. Is the following problem decidable?
Given a TM $M$ and $w$, does $M$ use finite number of tape cells for input $w$? (15)

For undecidability proof, you must describe the many-one reduction function precisely. Otherwise, the decision algorithm must be described in a structured manner.

If we know that a TM uses finite (bounded) space then we can decide if $M$ accepts $w$ or not. Even if the exact bound is not known, we can try iteratively space bound $1, 2, \ldots, i, \ldots$ which is a maximum of $2^{O(i)}$ steps. Then either
(i) $M$ accepts $w$ (ii) $M$ rejects $w$ by stopping in a non-accepting state or (iii) enters an infinite loop (and hence rejects $w$).

In cases (ii) and (iii), we can easily modify $M$ (by adding lines of code) such that subsequently the head moves right indefinitely.

The reduction function $f$ takes input $(M, w)$ and outputs $(M', w')$ that ignores its input $w'$ and simulates $M$ on $w$ as above. $M'$ accepts $w'$ if $M$ accepts $w$. Thus

$$w \in L(M) \iff M' \text{ uses finite space}$$

Moreover $M', w'$ can be computed by a recursive TM (that inserts the additional code to achieve the above behavior). Therefore we have achieved $L_u \leq L_{finite}$ using a Turing computable function.

If the proof is not written explicitly as a description of the many-one Turing computable reduction function and what exactly is the input-output of the reduction function, maximum marks given is 5.

The technique of many-one reduction is a fundamental concept that was emphasized in the course and it is important that it is demonstrated in the answers written.

It was iterated after last assignment submission and mentioned explicitly in the question. It is possible that your answers can be modified to this format but this was strictly enforced and accordingly penalized and there will be no re-evaluation if the description did not conform to this requirement.
4. A language $L_1 \leq \log L_2$ (LOGSPACE reducible) iff $x \in L_1 \iff f(x) \in L_2$ and $f(x)$ can be computed using a $\log |x|$ space-bounded turing machine with an additional write-only output tape where $f(x)$ is written and this space is not counted. Note that the length $|f(x)|$ can be much longer than $\log |x|$. The head on the output tape can only move right. Recall that it has a read only tape also whose space is not counted.

Show that if $L_1 \leq \log L_2$ and $L_2$ can be computed using a $\log n$ space bounded Turing machine, then $L_1$ can also be computed using a $\log n$ space bounded Turing Machine. Here $n$ is the length of the input. (15)

The straightforward approach is to integrate the TMs for the reduction function $f$ and $M_2$ on $f(x)$. $f$ uses $\log |x|$ storage tape and $M_2$ uses $\log |f(x)|$ space ($f(x)$ is the input to $M_2$).

But note that $|f(x)| \leq 2^{O(\log |x|)}$ which can exceed $\log x$ and therefore we cannot store the output tape of $f$ explicitly in the composite machine. The challenging issue is how do we store $f(x)$ if we want to run the TM for $L_2$ on $f(x)$.

Instead we compute $f(x)$ on demand. We store the current head position on the write-only tape also the symbol being currently scanned by the head on the (hypothetical) output tape of the TM computing the reduction function. Whenever, the $j$-th symbol is required we restart the TM for the reduction function and keep computing until the $j$-th output symbol is produced. Since the counter only needs to count till $|f(x)|$ which needs $\leq O(\log x)$ and we can reduce by a constant factor to make it less than $\log x$ (no constant factor).

Since $f(x)$ is the input to $M_2$, it cannot be modified, so generating the symbol scanned by the head will not affect the computation.

**Simply Composing the two machines does not work since the output tape of $M_1$ becomes an internal storage tape and therefore the space counts. This was the most common mistake and heavily penalized since this was the most important part of the construction**.
5. Given an undirected graph $G = (V, E)$ it is 3-colorable iff there is a coloring of the vertices $\chi : V \rightarrow \{1, 2, 3\}$ such that for all edges $(x, y) \in E$, $\chi(x) \neq \chi(y)$.

By assuming that the 3-coloring problem is NP-complete, prove that the 4-coloring problem is also NP-complete. Here the colors are from the set $\{1, 2, 3, 4\}$. (10)

Clearly, 4-coloring problem is in NP as we can verify whether the coloring is legal by checking all edges.

Now we want to achieve $L_{3\text{col}} \leq_f L_{4\text{col}}$ where $f$ can be computed in polynomial time. So, given a graph $G(V, E)$, we have to define $G'(V', E') = f(V, E)$ such that $G'$ is 4-colorable iff $G$ is 3-colorable.

$G'$ is defined as follows $V' = V \cup \{t\}$ and $E' = E \cup \forall v \in V, \{v, t\}$. Essentially $G'$ contains an additional vertex $t$ that is connected to all vertices.

If $G$ is 3-colorable then we can assign the 4th color to $t$ and therefore all edges will distinct colors on the two points. So $G'$ is 4-colorable.

If $G'$ is 4-colorable then $G$ must be 3-colorable - we can prove this by contradiction. Suppose not, then to color $t$ we will be forced to use a fifth color since $t$ is connected to all vertices, i.e., $G'$ is not 4-colorable.

If the answer was written without giving the explicit reduction (construction of $G'$) and using some algorithmic construction based on coloring then no credit was given. Every answer must include the proof that it is in NP.

After reduction, the proof must be done both ways - if and only if other reductions are possible but proof must have all the previous components.