Simrun Heir and Michael Sanchez
I pledge my honor that I have abided by the Stevens Honor System.
Problem 1 Show that the language L = {< M, w, k > : TMM accepts input w and never moves its head beyond the first k tape cells} is decidable.
Notes:

- Thm 5.9: A_LBA is decidable
    - M functions like an LBA but instead of only having w on the input tape, it can either act like a normal TM when k>w in length
    - If k<w then M is an LBA that has the first k symbols of w on its input string
- M has a finite number of configurations
    - Visits all cells from first cell to k-th cell
    - For it to not move past the k-th cell there must be some loop
        - Pigeonhole principle
- How M works
    - If w is accepted without moving past the first k tape cells, ACCEPT
    - If M moves past the first k tape cells, REJECTS
    - If M rejects, REJECT
- Either if the LBA accepts/rejects the language or if M accepts/rejects the language the language will be decided

Answer:
Assume that L is decidable and let TM K be the TM that decides language L.
Use K to construct a TM D that decides A_LBA.
D = "On input <L>

1. Construct a LBA F that accepts the same strings as M, but instead accepts or rejects them based on the first k symbols of the input (Note that when the length of k > w, the input for F will have a size < k, since it is unable to have blank symbols in its tape, thus always ensuring there are always <=k symbols being read by the tape head)
    a. On input <w> Run M
    b. If w is accepted, ACCEPT. Otherwise, REJECT.
2. Run K(F)
    a. If F accepted, ACCEPT
    b. Otherwise, REJECT."

This is a TM that utilizes the properties of an LBA, which is decidable, to modify the input length of the string w, and is able to decide the language L while never moving it's tapehead past the first k tape cells.

Problem 2 Recall that $A <_p B$ if there is a polynomial-time computable function f such that $w \in A \Leftrightarrow f(w) \in B$.
a. Show that the relation $<_p$ over languages is transitive.
b. Show that if $\forall A, B \in \boldsymbol{P}$, if $B \neq \phi$ and $B \neq \Sigma*$ then $A <_p B$. (Hint: this is easier than it seems since $A \in \boldsymbol{P}$. Now use the definition.)
c. Show that if $\boldsymbol{P}$ = **NP** then every language, other than ϕ and $\Sigma*$, in $\boldsymbol{P}$ is **NP**-Complete. (You can use the result of part (b) even if you didn't prove that.)

Notes:
Thm 7.31: If A ≤p B and B ∈ P, then A ∈ P.
Definition 7.29 : A ≤P B, if a polynomial time computable function f : Σ∗−→Σ∗ exists, where for every w, w ∈ A ⇐⇒ f(w) ∈ B.
a)
Have to show that if L1 $<p$ L2, and L2 $<p$ L3, then L1 $<p$ L3
b)

c)
Problem 7.18 in textbook
Thm 7.35: if B is NP-complete and B ∈ P, then P=NP
Thm 7.36: If B is NP-complete and B ≤p C for C in NP, then C is NP-complete.
Answer:
2a)
Let L1 $<p$ L2,
by definition there exists a polynomial-time computable function f such that w ∈ L1 ⇔ f(w) ∈ L2.
Let L2 $<p$ L3,
by definition there exists a polynomial-time computable function g such that x ∈ L2 ⇔ g(x) ∈ L3.
Thus, w ∈ L1 ⇔ f(w) ∈ L2 ⇔ g(f(w)) ∈ L3.
Note that the composition of two polynomials is also a polynomial.
This can be rewritten as w ∈ L1 ⇔ g(f(w)) ∈ L3, which, by definition means that L1 $<p$ L3.
Therefore, $<p$ has a transitive relation over languages.
2b)
Let A ∈ **P**.
Let ∀A, B ∈ **P**, and B ≠ ϕ and B ≠ Σ∗
Assume that A $<p$ B.
Since B ∈ **P**, and A ∈ **P**, then by definition A $<p$ B.
2c)
Let **P** = **NP**
A language A is said to be **NP**-complete if A is in **NP**, and every B in **NP** is polynomial time reducible to A
Let A be any language in **NP**.
Let B be any language other than ϕ and Σ∗.
Let B $<p$ A, which can be determined in polynomial time.
Then A ∈ **P**, and every B in **NP** is polynomial time reducible to A, A is **NP**-Complete.
Therefore, since any language in **NP** is **NP**-complete when **P** = **NP**, and any language in **NP** also belongs to **P**, all **P** is **NP**-Complete excluding ϕ and Σ∗

Problem 3 Behold, a genie appears before you! Given a formula $\phi(x_1, x_2, \dots, x_n)$ in conjunctive normal form with n boolean variables, the genie will correctly tell you (in one step) whether or not the formula is satisfiable. Unfortunately, the genie will not give you a truth assignment to the variables that makes the formula true. Your problem is to figure out a satisfying truth assignment when the genie says the formula is satisfiable. You can present the genie with a polynomial (in n) number of queries.

1. Give a high-level description of your algorithm, with enough detail.
2. What is the maximum number of queries made by your algorithm?
3. Explain why your algorithm correctly finds a satisfying assignment for a satisfiable formula.
4. A second genie appears! Given an undirected graph, this genie will correctly tell you whether or not the graph has a Hamiltonian cycle. How will you use this genie to find a Hamiltonian cycle in any graph that has one in polynomial time?

Notes:
3.4)
Thm 7.46: HAMPATH is NP-complete
From textbook, HAMPATH is decidable by a NTM
Thm 7.35: If B is NP-complete and B $\in$ P, then P=NP
Answer:
3.1) A exponential time algorithm M operates as follows:
M = “On input w = $\phi(x_1, x_2, \dots, x_n)$>:

1. Guess a distinct truth assignment and check with genie if it equals w
2. Repeat if genie doesn’t confirm truth assignment, otherwise ACCEPT”

3.2) $O(2^n)$
3.3) The algorithm will test every possible truth assignment until the genie accepts
3.4)
N_1 = “On input $\langle G, s\rangle$, where G is a undirected graph with node s:

1. Write a list of m numbers, $p_1,\dots,p_m$, where m is the number of nodes in G. Each number in the list is nondeterministically selected to be between 1 and m.
2. Check for repetitions in the list. If any are found, other than $p_1 = p_m$, reject.
3. Check whether s = p_1 and s = p_m. If either fails, reject.
4. For each i between 1 and m − 1, check whether (p_i, p_i+1) is an edge of G. If any are not, reject. Otherwise, all tests have been passed, so accept.”

Problem 4

a) Give a high-level description of a linear time algorithm to determine if a directed graph contains a directed cycle.
b) Next, suppose you are given a formula in 2CNF with n variables $x1, \dots, xn$. Construct a graph with 2n vertices, one for each literal, and for every clause construct two edges as follows:

If the clause is of the form ($xi \vee xj$), add directed edges ($\neg xi$, xj) and ($\neg xj$, xi).
If the clause is of the form ($\neg xi \vee xj$), add directed edges (xi,xj) and ($\neg xj$,$\neg xi$).
In general, for the clause is (a,b), add directed edges ($\neg a$,b) and ($\neg b$,a).
Describe how you would use your algorithm from part (a) to determine if the 2CNF formula is satisfiable and prove that your algorithm for satisfiability is correct.
Notes:

a) Could be constructed similar to HAMPATH, but instead just tries to get back to start node

Thm 7.14: PATH $\in$ P (can possibly use poly-time algorithm M for PATH as a model)

b)

Thm 7.27: SAT $\in$ P iff P = NP.
Answer:
4a)
A polynomial time algorithm M for D-CYCLE operates as follows.
M = “On input <G,s,t> where G is a directed graph with node s:

1. Place a mark on node s.
2. Repeat the following until either no additional node is marked or you end up back at s
    a. Scan all the edges of G from the currently marked node.
    b. If an edge (a,b) is found going from a marked node a to an unmarked node b, mark node b, and move to node b
        i. Note that starting from s makes s the first value for a, and after moving to the newly marked node b, b becomes a and the process is repeated
3. If t is marked, and t is the same node as s, the node that was started from, ACCEPT. Otherwise, REJECT.”

4b)
After constructing the graph with the specified method detailed in the problem, run D-CYCLE on the graph, identifying all cycles within the graph.
Examine each cycle