Problem 2

Claim: NP is not included in $DTIME(n^k)$ for any fixed $k \geq 1$.

Proof: This can be shown via a contradiction arising from application of the time hierarchy theorem. Assume that the claim is true, that is, there exists some $k$ such that $NP \subseteq DTIME(n^k)$. By corollary 5.15 of the time hierarchy theorem, $DTIME(n^k) \subseteq DTIME(n^{k+1})$. We know that $DTIME(n^{k+1}) \subseteq NTIME(n^{k+1})$ via the fact that $P \subseteq NP$. Therefore, $DTIME(n^{k+1}) \subseteq NTIME(n^{k+1}) \subseteq NP \subseteq DTIME(n^k) \subseteq DTIME(n^{k+1})$. A set cannot be properly contained within itself, so the claim must be false.
Problem 3

Claim: \( NSPACE(2^n) \subseteq DSPACE(2^{2n}) \)
Proof: By Savitch’s theorem, \( NSPACE(2^n) \subseteq DSPACE(2^{2n}) \). Since \( 2^{2n} \in o(2^{n^2}) \) and both functions are fully space constructible, we can apply the space hierarchy theorem and state that \( NSPACE(2^n) \subseteq DSPACE(2^{2n}) \subseteq DSPACE(2^{n^2}) \).

Problem 4

Claim: \( L = \{e | M_e \text{ accepts the string } 00 \} \) is c.e.
Proof: To show that a language \( L \) is c.e., it suffices to construct a TM \( N \) which will halt on on all strings \( x \) s.t. \( x \in L \), and not halt otherwise. Let \( N \) have a read-only input tape and three work tapes. Construct \( N \) as follows:

1. Read the input \( e \), which is assumed to be a valid encoding of a Turing Machine.
2. Use tape 1 for any operations necessary to construct simulate \( M_e \).
3. Simulate \( M_e(00) \) on the second work tape, allowing it to use the third work tape as its own work tape. Note that \( M_e \) may never halt, so \( N \) may never halt.
4. If \( M_e \) halts in an accepting state, halt and accept.
5. If \( M_e \) halts in a rejecting state, loop.

Note that the above TM will always halt if \( M_e \) accepts the string 00, and will never halt otherwise, therefore \( L \) is acceptable. Since a set if c.e. if and only if it is acceptable, \( L \) is c.e.

Problem 5

Claim: Any partial c.e. set is actually a c.e. set.
Proof: A set is defined as partial-c.e. if it is the range of a partial-computable function. To prove the claim, we can show (via Corollary 3.2) that any partial-c.e. set is also the domain of a partial computable function. For partial-c.e. set \( S \), \( \exists f \text{ s.t. } range(f) = S \) and \( f \) is partial-computable. By definition, there also exists a TM \( M \) which computes \( f \). Then we can construct an algorithm which computes the partial-computable function \( g : N \rightarrow N \) s.t. \( domain(g) = S \). The algorithm works as follows:

1. On input word \( w \):
2. \( x = 1 \)
3. While (true)
   (a) Start simulating \( M(x) \)

2
(b) Evaluate one step of all currently running computations. If any halt with $w$ on
their output tape, halt and accept with $x$ on the output tape.

(c) Increment $x$

Note the above algorithm halts only if $f(w)$ is defined. That is, only if $w \in \text{range}(f)$. Furthermore, note that if the algorithm halts, then $w$ is by definition in the domain of $g$, since $g$ is defined (halts with an output) on $g(w)$. If one of the above statements is not true, then the algorithm does not halt. $S = \text{domain}(g)$ and $S = \text{range}(f)$, therefore parti-c.e. set $S$ is c.e.