Section 8

Problem 1

Function F : $\{0,1\}^* \to \{0,1\}$ checks whether the input encodes a TM that, on every input for which it halts, outputs either a string with at most n 0s or a string with length at least n. Prove that F is uncomputable using Rice's theorem or state why Rice's theorem does not apply and show polynomial time algorithm.

Solution Rice's Theorem does not apply, because a property is secretly trivial. This is because if the output string has more than n 0s its length is also longer than n. Therefore, for every input the returned value should be 1, which we can trivially implement in polynomial time using Python.

Problem 2

Prove that if F, G : $\{0,1\}^* \to \{0,1\}$ are in P then their composition F $\circ$ G, which is the function H s.t. H(x) = F(G(x)), is also in P.

Solution Lemma: First, note that the composition of two polynomial functions is polynomial: the composition of two functions f and g with degrees d1 and d2 respectively can have a maximum degree of $d_1 * d_2$, which is still polynomial.

For the main proof, note that if F and G are in P, then there must be NAND-RAM programs PF and PG that compute F and G respectively that run for $O(n^{k_1})$ and $O(n^{k_2})$ steps of NAND-RAM respectively, where n is the length of the input.

By the sequential composition theorem, we can construct a program PC that computes F(G(x)) in the following way: Start with the program PG, and compute G(x). Then "paste in" the code for PF , changing the input to PF to the output G(x) from PG. Note that because F is bounded by $O(n^k)$ NAND-RAM steps, it can be written as a NAND program of $O(poly(n^{k_1}))$ lines, which means the size of the output of PG can be at most $poly(n^{k_2})$. By our lemma, this upper bound on the size of the output is still polynomial, so let the size of the output of the polynomial be $O(n^c)$ for some constant c. This means that the length of the input to the code for PF is at most $O(n^c)$, so the overall runtime is $O((n^c)^k) = O(n^{ck})$, which is still polynomial.

Problem 3

Prove or disprove: F is uncomputable. Let F be the following function. On input a (string representing a) pair (M, P) where M is a Turing Machine and P is a NAND-TM program, F outputs 1 if and only if M and P are functionally equivalent, in the sense that for every $x \in \{0,1\}^*$, either both M and P don't halt on x, or M(x) = P(x).

Solution Assume for contradiction that F is computable. Reduce HALT to F.

Suppose a program T computes F. We construct a program PHALT that computes HALT. Given a NAND-TM program P, create P' by adding line return 0 at the end of the program.

Now we show we can compute HALT(Q), by creating program PHALT(P). This program:

1. creates P' as described above
2. creates M to be a TM that just returns 0
3. returns T (M,P').

Suppose that P halts, then P' halts and returns 0. Since M also halts and returns 0, T(Q,M) = 1 and PHALT returns 1, as desired.

Suppose that P does not halt, then P does not halt so T(Q,M) = 0. PHALT returns 0, as desired.

Therefore we can compute HALT, which is a contradiction! Hence, F is uncomputable.

Other version of the proof uses Rice's Theorem