Name: Key **CS301 F10**

1. Consider the problem of determining whether a Turing machine does not use all $\gamma \in \Gamma$ on its tape when it is run on input w.

   (a) (__ /1 pt) Formulate this problem as a language.

   $C = \{ \langle M, w \rangle \mid M$ is TM and does not use some $\gamma \in \Gamma$ on tape when given $w. \}$

   (b) (__ /2 pts) How would the proof that the language above is undecidable differ from the proof, on homework, of the undecidibility of

   $B = \{\langle M, w\rangle \mid M$ 2-tape TM that writes $\gamma \neq \sqcup$ on 2nd tape given $w\}$?

   use w/ decider for $\overline{C}$. →

   T = "On input x:
   1. Simulate M' on x
   2. If simulation shows that M' accepts, write all $\gamma \in \Gamma$ on tape.

   Where M' is M, except all $\gamma \in \Gamma$ are replaced with γ'.

2. Let $R_{\text{TM}} = \{\langle M\rangle \mid M$ is a TM and $L(M)$ is regular$\}$. Consider, from homework, the proof that R_{TM} is undecidable, and answer the following.

   (a) (__ /2 pts) M_2 is designed to first accept input x from the language $0^n 1^n$ since, critically, that language is not regular.

   (b) (__ /2 pts) Consider $C_{\text{TM}} = \{\langle M\rangle \mid M$ is TM and $L(M)$ is not CFL$\}$. How can you change this proof to instead show that C_{TM} is undecidable?

   - Use a decider for $\overline{C_{TM}}$ (which is C_{TM} on your Q10.)
   - Change $0^n 1^n$ to non-CFL language, as on Q10.
     e.g. $0^n 1^n 2^n$.

3. Let $Z_{\text{CFG}} = \{\langle G\rangle \mid G$ is CFG and $L(G) = \Sigma^*\}$. Consider, from homework, the proof of Theorem 5.13: *Z_{CFG} is undecidable.*

   (a) (__ /1 pt) **TRUE/FALSE:** Every non-deterministic PDA can be converted to an equivalent CFG. *You may need to check your textbook...*

   Thm 2.20 (pg. 117)

   (b) (__ /2 pts) Can you change the "proof" so that it works using a deterministic PDA? *Answer on the back of this page.*

   check start configuration

   $\# C_1 \# C_1 \# C_2^R \# C_2 \# C_3^R \# \ldots \# C_{\ell-1} \# C_\ell^R \# C_\ell$

   check each $C_{i-1} \to C_i$

   check accepting configuration

   NOTE: Can process deterministically from left to right.