(1) A language is *linear* if it is the language of a context-free grammar in which every production (rule) has at most one variable on the right-hand side – that is, a *linear grammar*. A *palindrome* is a string for which $w = w^\text{rev}$. Define $\text{PAL}_{\text{LIN}} = \{ \langle G \rangle : G \text{ is a linear grammar for which } L(G) \text{ contains at least one palindrome} \}$. Prove or disprove: $\text{PAL}_{\text{LIN}}$ is decidable.

(2) Define $\text{PAL}_{\text{REX}} = \{ \langle R \rangle : R \text{ is a regular expression for which } L(R) \text{ contains at least one palindrome} \}$. Prove or disprove: $\text{PAL}_{\text{REX}}$ is decidable.

(3) Often we talk about languages like $\text{P}_{\text{TM}} = \{ \langle M \rangle : M \text{ is a TM for which } L(M) \text{ has property P} \}$. Let’s define $\neg \text{P}_{\text{TM}} = \{ \langle M \rangle : M \text{ is a TM for which } L(M) \text{ does not have property P} \}$. For the types of TM encodings we outlined in class, it is not the case that $\neg \text{P}_{\text{TM}} = \overline{\text{P}_{\text{TM}}}$ because $\overline{\text{P}_{\text{TM}}}$ contains strings that do not encode Turing machines at all. Nevertheless, to determine whether $\neg \text{P}_{\text{TM}}$ is recognizable or decidable, we often instead just determine whether $\overline{\text{P}_{\text{TM}}}$ is recognizable or decidable. Explain carefully why we are justified in doing this.

(4) An *unhappy* Turing machine is one that accepts every string except for its own description. Show that there are infinitely many unhappy Turing machines.

(5) It may happen that, although a language $L$ is not Turing-recognizable, some infinite subset $L' \subseteq L$ is recognizable. For example, the language $\{ \langle M, w \rangle : M \text{ does not accept } w \}$ is not recognizable, but the infinite sub-language $\{ \langle M, w \rangle : M \text{ halts and rejects } w \}$ is recognizable. Establish that there is a language $L$ for which no infinite subset of $L$ is recognizable. (Hint: the recursion theorem will be useful.)