0. (1 point) Submit your homework in accordance to the guidelines at
https://www.cc.gatech.edu/~rpeng/CS4510_S20/HomeworkGuidelines.pdf.

1. (8 points) A language is undecidable if there is no Turing Machine that will recognize that
language and halt on all inputs. Show that the following language is undecidable:

\[ A = \{ M \mid M \text{ is a Turing machine that accepts exactly all the odd length strings} \} \]

The intended solution is to reduce from the halting problem, which we will show is undecidable.
The halting problem is given a Turing machine \( M \) and an input \( w \), decide whether \( M \) terminates
when given \( w \) as input.

**SOLUTION:**
We reduce the halting problem to deciding \( A \). Recall

\[ \text{HALT} = \{ \langle M, w \rangle \mid M \text{ terminates when ran on input } w \} \]

Let \( < M, w > \) be a TM / input pair. That is, we want to decide whether \( M \) halts on \( w \).

Construct \( M' \) that on any input \( s \), simulates \( M \) on input \( w \) (we can hard code \( w \) into the
construction of \( M' \)). If \( M \) halts \( w \) (either accept or reject), then \( M' \) checks whether \( s \) has odd
length, and accepts \( s \) if it has odd length.

Then:

(a) If \( M \) does not terminate on \( w \), then \( M' \) accepts nothing.
(b) If \( M \) halts on \( w \) (either accepts or rejects), then \( M' \) accepts all odd length strings.
In other words, $M' \in A$ if and only if $< M, w > \in \text{HALT}$, what we constructed is a valid reduction from $\text{HALT}$ to $A$. As $\text{HALT}$ is undecidable, $A$ is undecidable as well.

**GRADING:**

(a) 1 for reducing from an undecidable problem  
(b) 1 for picking explicit $M, x$ for undecidable problem  
(c) 2 for correct direction of reduction  
(d) 2 for reduction holds for both halves of “iff” statement  
(e) 1 for making new machine $M'$ which runs $M$ inside of it  
(f) 1 for no minor errors

2. (8 points) Consider the ambiguous CFG problem: given a context free grammar, does there exist a string that can be generated by two different parse trees. Show that this problem is undecidable by reducing the post correspondence problem to it.

The post correspondence problem is the following: given two sequences of strings $a_1, ..., a_n$ and $b_1, ..., b_n$, is there a sequence of indices $i_1, ..., i_m$ (with possibly repeats, and whose total length may be much bigger than $n$) such that $a_{i_1}a_{i_2}...a_{i_m} = b_{i_1}b_{i_2}...b_{i_m}$?

**SOLUTION:**
Let the strings be $a_1 ... a_n, b_1 ... b_n$. We construct a CFG with rules:

\[
S \rightarrow S_1 \mid S_2 \\
S_1 \rightarrow iS_1 a_i \\
S_2 \rightarrow iS_2 b_i \\
S_1 \rightarrow \# \\
S_2 \rightarrow \#
\]

Then if we start with $S_1$, the left part (with the indices) uniquely determines the right portion, as $\#$ is a unique symbol.

Similarly, everything generated from $S_2$ is unique. Thus the only ambiguity comes from $S_1$ and $S_2$ generating the same string, but as the indices are the same, the portions right of the $\#$ must agree as in the post correspondence problem.

3. (9 points) Show that if a language $L$ is in Turing recognizable, then $L^*$ is also Turing recognizable.

Note that unlike Turing machines that *decide* languages, Turing machines that *recognize* languages may not halt on inputs not in the language. Formally, a Turing machine $M$ *recognizes* a language $L$ if for every string $w \in L$, $M$ terminates with accept when given $w$ as input.
SOLUTION:
Let a Turing machine that recognizes $L$ be $M$.
Given a word $w$, we take all possible decomposition of $w$:

$$w = w_1w_2 \ldots w_k.$$ 

We then check in parallel whether $M$ accepts $w_1 \ldots w_k$ in each of the breakdown. As soon as $M$ accepts one of the breakdowns, we accept.

If $w \in L$, then some breakdown will have each of its strings be accepted by $M$. So running $M$ on each of the strings sequentially will all terminate with accept everywhere, and thus accept in that run on the breakdown. As we check all the splits in parallel, this means that $w$ will be accepted as well.

GRADING:

(a) 2 points for saying there is a TM that recognizes $L$.
(b) 2 points for enumerating all breakdowns (ideally this should come with a description of how it’s implemented on TM).
(c) 1 point for the right accept/reject conditions (accept iff one of the breakdowns accept)
(d) 2 points for running things in parallel.
(e) 2 points for justifying termination if any of the breakdowns terminate, aka. properly handle the case where $M$ loops forever on one of the (earlier) breakdowns.