1. [10 pts] Let $A101-CFG = \{< G >| \, G \text{ is a CFG that generates string } 101\}$, prove that $A101-CFG$ is decidable.

Run G on input w:

- **We can make a PDA C that accepts the string 101.**
- **Next, we have to make a PDA D that accepts any context free string.**
- **We can then intersect the 2 PDAs so L(C) intersected with L(D) and name it L(E)**
- **We run TM R on E which is the ECFG**
    - **If R accepts, then G rejects**
    - **If R rejects, then G accepts**
- **Decidable ===**

2. [10 pts] Let $B = \{< D >| \, D \text{ is a DFA that accepts } \omega \text{ whenever it accepts the reverse of } \omega\}$.
Prove that B is decidable.

Run D on input w:

- **A DFA that accepts only palindromes is context free but not regular.**
- **We can make a PDA J that accepts w wherever it accepts the reverse of w.**
- **We will also make another DFA K that accepts any string**
- **Although DFA K is regular, both PDA J and DFA K are context free and so they are closed under the intersection operation**
- **We can intersect L(J) and L(K) and the result we can call V**
- **We TM R or a emptiness decider on V**
    - **If V accepts, then D rejects**
    - **If V rejects, then D accepts**
- **Decidable ===**

3. [20 pts] Given a Turing machine M, we consider the problem of deciding whether this Turing machine has an equivalent finite automaton or not, formulate this problem as a language, then prove that it is undecidable.

X = {<M> | <M> is a turing machine and has a equivalent finite automaton}

- **Because we are checking if equivalent finite autonomous, we are effectively checking if the TM accepts regular languages.**
- **We can use REGTM and do a proff by reduction**
- **First, We have to assume that REGTM and ATM are both decidable:**
- **Let V be a REGTM decider**
- **Make V construct TM S which is a ATM decider**
- **TM M on input w and M' on input x**
    - **If x is in the form {w = w^r | w is a palindrome}, accept**
    - **If x is not in a palindrome, Run TM M on input w and accept if M accepts**
- **Run V on input <M'>**
    - **If R accepts, then accept**
    - **If R rejects, then reject**
- **Since we know ATM is undecidable but proved that it is, our initial assumption is incorrect**
- **Undecidable ==**

4. [20 pts] Let G1 and G2 be two CFGs, formulate the problem of deciding whether G1 and G2 generate the same set of strings as a language. Then prove the problem is undecidable.
J = {<G1, G2> | G1 and G2 are both CFGs and L(G1) = L(G2)}

- **Let V be a turing machine that decides J and then construct another TM S to decide ALLCFG**
- **TM S on input <G> where G is a CFG:**
- **Run R on input <G, G3> where G3 is a CFG that accepts any string**
    - **If R accpets, then accept,**
    - **If R rejects, then reject**
- **Since we know that ALLCFG undecidable but proved that it is, then we know that are initial assumption is wrong**
- **Undecidable ==**

5. [20 pts] Let $SUBCFG{-}DFA = \{\langle G, D\rangle \mid G \textit{ is a CFG and } D \textit{ is a DFA and } L(G) \subseteq L(D)\}$. Is $SUBCFG{-}DFA$ decidable? If your answer is yes, describe a TM that decides it; if your answer is no, prove it.

- **If we intersect a CFL and a RL the result will be a CFL.**
- **L(D) is regular and L(G) is context free language**
- **On input <G, w>:**
- **Convert G into a CNF and get all derivations which would be 2w-1. Intersect G and D and the result is K.**
- **Run TM R on K:**
    - **If K accepts, accept**
    - **If K rejects, reject**
- **Decidable ==**

6. [20 pts] Given a CFG G, we consider the problem of determining whether G generates any palindromes or not. Formulate this problem as a language, then prove whether it is decidable or not.

- **A DFA that accepts only palindromes is context free.**
- **We can make a PDA C that accepts w wherever its a palindrom**
- **We will also make another PDA D that accepts any string that is generated by a context free grammar**
- **Both PDA C and D are context free and so they are closed under the intersection operation**
- **D’ is everything that is not in the L(G)**
- **We can intersect L(C) and L(D’) and the result we can call E**
- **We TM R or a emptiness decider on E**
    - **If E accepts, then G accepts**
    - **If E rejects, then G rejects**
- **Decidable ==**