I. (10 pts.) Design a context-free grammar for the language 
\{xy | x, y \in \{a, b\}^*, |x| = |y|, x \neq y^R, and |x| is an odd integer\}.

\begin{align*}
S &\rightarrow \text{odd}, \ S_1 \rightarrow \text{even (before mismatch)} \\
A &\rightarrow \text{odd}, \ A_1 \rightarrow \text{even (after mismatch)} \\
X &\rightarrow a | b \\
\{ &\rightarrow XSX \\
S_1 &\rightarrow XSX \\
S &\rightarrow aA_1b | BA_1a \quad \text{(mismatch)} \\
S_1 &\rightarrow aA_1b | BA_1a \quad \text{(mismatch)} \\
A &\rightarrow XA_1X \\
A_1 &\rightarrow XAX | c
\end{align*}
II. (10 pts.) Specify a decision procedure for the following problem:

- Given \( n \) dfas, \( M_1, M_2, \ldots, M_n \), over the alphabet \( \Sigma \), do there exist 2 dfas \( M_k \) and \( M_\ell \), \( k \neq \ell \), such that \( L(M_k) \cup L(M_\ell) = \Sigma^* \)?

**Method 1:**
- Design a dfa for the language \( \Sigma^* - (L(M_i)) \cup L(M_j) \).
- Construct an rfa for \( L(M_i) \cup L(M_j) \).
- Convert it to dfa.
- Swap final and non-final states.
- Check whether there exists a path from the initial state to a final state.

**Method 2:**
- First argue that if \( L(M_i) \cup L(M_j) \neq \Sigma^* \) then:
  - For an input string of length \( \leq \beta_i, \beta_j \) (where \( \beta_i \) and \( \beta_j \) are the number of states of \( M_i \) and \( M_j \), respectively).
  - Feed every string of length \( \leq \beta_i, \beta_j \) into \( M_i, M_j \).
- If some string is rejected by both \( M_i, M_j \), then \( L(M_i) \cup L(M_j) \neq \Sigma^* \).

**Overall procedure**
- For every pair \((i, j)\) check whether \( L(M_i) \cup L(M_j) \neq \Sigma^* \).
- If for some pair the check succeeds respond "yes".
- Else respond "no" and halt.

**Method 0:**
- Design a dfa for the language \( L(M_i) \cup L(M_j) \).
- Construct an rfa for \( L(M_i) \cup L(M_j) \).
- Convert it to dfa.
- Minimize the number of states. \( \forall s \in \Sigma \),

\[
\text{If the minimized dfa is } \Sigma^*, \ldots
\]

\[
L(M_i) \cup L(M_j) = \Sigma^*.
\]
III. (10 pts.) Design a Turing machine for computing the following function:

\[ f(x, y, z) = \begin{cases} 
2x + y - 2z & \text{if } 2x + y - 2z \geq 0 \\
0 & \text{otherwise}
\end{cases} \]

- \( x \rightarrow 2x \) by shifting left \( x \) appending '0' on the right
- \( 2x + y \): by repeatedly subtracting 1 from \( y \) and adding 1 to \( x \)
- \( z \rightarrow 2z \): by appending 0,0,1 on the right
- \((2x+y)-2z\): by repeatedly subtracting 1 from \( y \) and \( x \) 

A to B: \( x \rightarrow 2x \)
B to C: \( 2x+y \)
C to D: \( z \rightarrow 2z \)
D to E: \( 2x+y-2z \)
D to F: 0
IV. (10 pts.) Solve one of the following problems.

(a) Assume that testing whether a given TM halts on exactly one input is undecidable. Prove that testing whether a given TM halts on exactly two inputs is undecidable.

(b) Recall that a given R-PCP \((x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\) has a solution if there is a sequence of indices \(i_1, i_2, \ldots, i_m, m \geq 1\), such that \(x_{i_1} x_{i_2} \ldots x_{i_m} = y_0 y_1 y_2 \ldots y_m\). Prove that the R-PCP problem is undecidable assuming that the PCP problem is undecidable.

(a) We show 1-input problem \(\leq_m\) 2-input problem

Let \(M\) be a TM that halts on exactly one input if \(M\) halts on exactly 2 inputs.

Transform \(M\) to \(M'\):
- \(M'\) has an input \(i\).
- If \(i = 0\), \(M'\) simulates \(M\) on \(i\) and \(i+1\) by
  - Overwriting \(M\) on at least one of those, \(M'\) halts.
- If \(i = 0\), \(M'\) simulates \(M\) on \(0\).

Correctness:
- If \(M\) halts on 1 input, \(M'\) halts on 1 input.
- If \(M\) doesn't halt on 1 input, \(M'\) doesn't halt on 1 input.

Also the transformation from \([A]\) to \([B]\) is computable.

(b) We show \(\text{PCP} \leq_m \text{PCP}\)

Transform \(E\) to \(E'\) such that \(E\) has a solution iff \(E'\) has a solution.

\[ E' = \{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\} \]

For each \(E' = \{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}\) in \(E\) introduce

\[ (c, c_1), (c_2, c_3), \ldots, (c_n, c_{n+1}) \]

Hence \(E'\) has 1 + 2m pairs.

Argue the correctness.