1. Draw a deterministic Turing machine that accepts $\{wcwc \mid w \in (a \cup b)^*\}$. For example, abbcabbbabb. [13 points]
2. Draw a deterministic Turing machine that accepts \{strings over alphabet \{a,b,c\} such that the number of a’s, the number of b’s, and the number of c’s are all equal\}. For example, abbcccbbaa. [13 points]
3. Does each of these instances of Post's Correspondence Problem (PCP) have a match? Either show a matching sequence or explain why no match exists. [12 points]

a. \[
\begin{bmatrix}
  a \\
  baa
\end{bmatrix}, 
\begin{bmatrix}
  baa \\
  a
\end{bmatrix}, 
\begin{bmatrix}
  ba \\
  baa
\end{bmatrix}.
\]

No solution.
Any matching sequence must begin 3, 1, 2, 1, 2, 1, ... or 3, 1, 3, 1, ... 
In each case, the top string will always remain shorter than the bottom string.

b. \[
\begin{bmatrix}
  a \\
  aab
\end{bmatrix}, 
\begin{bmatrix}
  aaa \\
  b
\end{bmatrix}, 
\begin{bmatrix}
  ba \\
  a
\end{bmatrix}.
\]

No solution.
Any matching sequence must begin 1, 1, 3, 1, 3, 1, 1, 3, ... 
The top string will always remain shorter than the bottom string.

c. \[
\begin{bmatrix}
  abb \\
  a
\end{bmatrix}, 
\begin{bmatrix}
  b \\
  abb
\end{bmatrix}, 
\begin{bmatrix}
  a \\
  bb
\end{bmatrix}.
\]

Solution = 1, 3, 1, 1, 3, 2, 2.
4. The Input/Output (I/O) problem is defined as follows: Given a deterministic program in your favorite high-level programming language (such as C or C++ or Java) and an integer value X, does there exist any sequence of input values that will cause the given program to output the value X? Answer each question below, and justify each answer. [12 points]

a. Is the I/O problem decidable?

   No, by reduction from Empty$_{TM}$ problem to I/O problem. Apply Church’s thesis to convert a TM to a program P. Then modify the program P to write output $X = 1$ whenever it halts and accepts, otherwise no output.

b. Is the complement of the I/O problem decidable?

   No, because decidable problems are closed over complement.

c. Is the I/O problem Turing-recognizable?

   Yes. Let $<P, X>$ be input for I/O problem. For $k = 1, 2, 3, \ldots$, simulate running P on each input w whose encoding $<w>$ has at most k bits, for up to k time units per input. If X is ever written to the output, then accept.

d. Is the complement of the I/O problem Turing-recognizable?

   No, because if both I/O and its complement $\overline{I/O}$ were Turing-recognizable, then both must also be decidable, which is a contradiction.
5. Define a homomorphism to be any function $H: \Sigma_1 \rightarrow \Sigma_2^*$, where $\Sigma_1$ and $\Sigma_2$ are alphabets. Determine whether each claim below is true or false, and justify each answer. [10 points]

a. The Turing-recognizable (or recursively enumerable) languages are closed over homomorphism.

   True. Here is the solution available online for textbook exercise 9.2.6(e) part (i):

   Consider the case where $L$ is RE. Design a NTM $M$ for $H(L)$, as follows. Suppose $w$ is the input to $M$. On a second tape, $M$ guesses some string $x$ over the alphabet of $L$, checks that $H(x) = w$, and simulates the TM for $L$ on $x$, if so. If $x$ is accepted, then $M$ accepts $w$. We conclude that the RE languages are closed under homomorphism.

b. The decidable (or recursive) languages are closed over homomorphism.

   False. Here is the solution available online for textbook exercise 9.2.6(e) part (ii):

   Consider the particular language $L$ consisting of strings of the form $(M,w,c')$, where $M$ is a coded Turing machine with binary input alphabet, $w$ is a binary string, and $c'$ is a symbol not appearing elsewhere. The string is in $L$ if and only if $M$ accepts $w$ after making at most $i$ moves. Clearly $L$ is recursive; we may simulate $M$ on $w$ for $i$ moves and then decide whether or not to accept. However, if we apply to $L$ the homomorphism $H$ that maps the symbols other than $c$ to themselves, and maps $c$ to $\varepsilon$, we find that $H(L)$ is the universal language, which we called $\text{Accept}_{TM}$ (or $L_u$). We know that $\text{Accept}_{TM}$ is not recursive.