1) a) We show L is not recursive by a reduction from the (non-recursive) membership problem. Define $f(M\#x) = M\#xx$. Then $M\#x$ in MP iff M accepts x iff $M\#xx$ in L (since xx consists of two strings of 0s and 1s of equal length, so only the condition M accepts x remains) iff $f(M\#x)$ in L.

Note that L is r.e. (recursively enumerable) as can be shown e.g. by a machine that first checks whether the input has the right shape (can be split into two parts of the same length, i.e. has even length) and then simulates running the TM on the first half of the input (by means of the/a universal TM).

b) MP and HP are clearly distinct languages (for a TM M and word x such that M rejects x, e.g. for a TM M that immediately goes into the reject state, we have $M\#x$ in HP but not in MP), but as shown in the lecture each can be reduced to the other. The languages in an equivalence class can all be reduced to each other (and are all such).

2) • The intermediate matrices and the solution are:

$$\begin{pmatrix} 0 & 4 & 3 & 1 \\ 1 & 0 & \infty & 3 \\ \infty & 1 & 0 & \infty \\ \infty & \infty & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 4 & 3 & 1 \\ 1 & 0 & 4 & 2 \\ \infty & 1 & 0 & \infty \\ \infty & \infty & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 4 & 3 & 1 \\ 1 & 0 & 4 & 2 \\ 2 & 1 & 0 & 3 \\ \infty & \infty & 1 & 0 \end{pmatrix} \Rightarrow$$

$$\begin{pmatrix} 0 & 4 & 3 & 1 \\ 1 & 0 & 4 & 2 \\ 2 & 1 & 0 & 3 \\ 3 & 2 & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 3 & 2 & 1 \\ 1 & 0 & 3 & 2 \\ 2 & 1 & 0 & 3 \\ 3 & 2 & 1 & 0 \end{pmatrix}$$

• The computation of the spanning forest is:

k_i	$b(k_i)$	P
		$\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\}\}$
$\{a,b\}$	1	$\{\{a,b\},\{c\},\{d\},\{e\},\{f\},\{g\}\}$
$\{b,f\}$	1	$\{\{a,b,f\},\{c\},\{d\},\{e\},\{g\}\}$
$\{c,g\}$	1	$\{\{a,b,f\},\{c,g\},\{d\},\{e\}\}$
$\{a,e\}$	2	$\{\{a,b,e,f\},\{c,g\},\{d\}\}$
$\{c,d\}$	2	$\{\{a,b,e,f\},\{c,d,g\}\}$
$\{e,f\}$	2	
$\{b,e\}$	3	
$\{d,g\}$	3	

The final computed spanning forest is:

$$\{\{a,b\},\{b,f\},\{c,g\},\{a,e\},\{c,d\}\}$$

3) a) The Chinese Remainder Theorem tells us, how to find for any integer x, for which we

know the rest of the euclidean division for two positive and relatively prime integers p, q, the remainder of x by the division of $p \cdot q$ and that this solution is unique.

b) We see, that $-1 \cdot 7 + 1 \cdot 8 = 1$ (alternatively we can compute that with Bezout's Lemma). We choose $a = 5, p = 7, b = 2, q = 8, u = -1$ and $v = 1$ of the Chinese Remainder Theorem and we get $x = 1 \cdot 8 \cdot 5 + -1 \cdot 7 \cdot 2 = 26 \mod 56$ as a solution for the first congruence system. For the second we only need to change the values of a and b to 3 resp. 4 and we get $y = -4 = 52 \mod 56$.

We verify our results for the first system: $26 = 3 \cdot 7 + 5 = 5 \mod 7$ and $26 = 3 \cdot 8 + 2 = 2 \mod 7$. For the second system: $52 = 7 \cdot 7 = 3 \mod 7$ and $52 = 6 \cdot 8 + 4 = 4 \mod 8$.

c) The congruence system

$$x = 2 \mod 4$$

$$x = 3 \mod 6$$

does not have a solution. If it would have a solution the congruence system

$$x = 2 \mod 2$$

$$x = 3 \mod 2$$

would have one as well (Generally: If $c = d \mod wz$, then $c = d \mod w$, since if $c - d$ is a multiple of wz, then it is so of w as well). $x = 0 \mod 2$ and $x = 1 \mod 2$ cannot hold at the same time, since a number cannot be even and odd at the same time. This does not contradict the Chinese Remainder Theorem, since 4 and 6 are not relatively prime, i.e., $gcd(4, 6) = 2 \neq 1$.

However, simply from the fact that $\gcd(4, 6) \neq 1$ we can not conclude that there is no solution, since the Chinese Remainder Theorem does not state anything about that case. In fact, there exist congruence systems that have a solution despite having $\gcd(p, q) \neq 1$. For example, just replace 3 by 4 in the system above.

4*) We first show L is not recursive by a reduction similar to the one in 1) but now from the halting problem.[1] Define $f(M\#x) = M'\#x$ where M' is constructed from M such that given an input M is run on its first half.[2] Then $M\#x$ in HP iff M halts on x iff M' halts on xx iff $M'\#x$ in L iff $f(M\#x)$ in L. Since the recursive languages are closed under complement, $\sim L$ is not recursive either.

[1] That is a bit more convenient here, since now L is defined via the TM halting instead of accepting as in 1).
[2] Though awkward to program at TM level, such an f is computable.