Question 1

Let look at the sentence “if cond then if cond then a :=1 else a:=1” This sentence can be created using two different trees

- Method 1

$$
\begin{gathered}
< STMT > \to < IF - THEN > \\
\to if\ cond\ then < STMT > \\
\to if\ cond\ then < IF - THEN - ELSE > \\
\to if\ cond\ then\ if\ cond\ then < STMT > else < STMT > \\
\to if\ cond\ then\ if\ cond\ then < ASSIGN > else < ASSIGN > \\
\to if\ cond\ then\ if\ cond\ then\ a \coloneqq 1\ else\ a \coloneqq 1
\end{gathered}
$$

- Method 2

$$
\begin{gathered}
< STMT > \to < IF - THEN - ELSE > \\
\to if\ cond\ then < STMT > else < STMT > \\
\to if\ cond\ then\ < IF - THEN > else < ASSIGN > \\
\to if\ cond\ then\ if\ cond\ then < STMT > else\ a \coloneqq 1 \\
\to if\ cond\ then\ if\ cond\ then < ASSIGN > else\ a \coloneqq 1 \\
\to if\ cond\ then\ if\ cond\ then\ a \coloneqq 1\ else\ a \coloneqq 1
\end{gathered}
$$

As we can see there is two tree to create this expression so the language is ambiguous

$< STMT > \to\ < ASSIGN > | < IF - THEN > | < IF - THEN - ELSE >$

$< IF - THEN > \to\ \ if\ cond\ then < STMT >$

$< IF - THEN - ELSE > \to if\ cond\ then < ASSIGN > else < STMT >$

$< ASSIGN > \to a \coloneqq 1$

Question 2

We can make a NPDA where each time it read a letter it can do two things: add this letter to the stack and continue on this state or go to the state corresponding to that letter (One state for each letter). Now the pda will in fact check if this letter(which is the nth in the word: stack of size n-1) is the same in the second word and if we see that the nth letter of the second word is not the same then we succeed. So we can now continue and guess when we have reach the half of the word and after that start poping out of the stack to get the nth letter if its not the same then we are good and we just check that the half choosen was really the middle by continuing to the end of the word

Question 3

Let’s take the language $(abc)^*$ then $perm(L)$ is all the words compose of the same number of a,b and c.

Let's now take the intersection of $perm(L)$ and the language $a^*b^*c^*$ then we get the language with all the words of the form $a^nb^nc^n$ but we know this language is not context free. So all words in L cannot be formed using a cfg so the language $perm(L)$ is not context free even if L is context free. Then in general if the language L is context free the langauge $perm(L)$ need not to be context free

Question 4

Given a word w we can generate all permutation of w then check for all permutation if at least one belongs to L and as L is recursive then the algorithm will always finish. As $perm(L)$ contains all permutation of all words in L then if we take all the permutation of one word and one of those permuation belongs to L then this word also belongs to L. Similarly if none of the permutation belongs to L then it means not word in L can generate a permutation equal to this word so the word doens belongs to $perm(L)$

Question 5

a)False

Let suppose we have a language L such that L is not R. Now let the language $L_w = \{w\}$ which is a language compose of only one word in L. Now we see clearly that L_w is R and also that $L = \bigcup_{w\in L} L_w$. Then L is the union of infinitely R language but is not R

b) False

Let suppose we have a language L such that L is not RE. Now let the language $L_w = \{w\}$ which is a language compose of only one word in L. Now we see clearly that L_w is RE and also that $L = \bigcup_{w\in L} L_w$. Then L is the union of infinitely RE language but is not RE

Question 6

2. Let take the language $\overline{L}$ and we know $\overline{L}$ is not RE so we know take the function $f(w) = bw$ But if we use f on $\overline{L}$ then all $f(w)$ are in K, similarly if w is not in $\overline{L}$ then $f(w)$ is not in K either. So we have $\overline{L} \leq_f K$. But as $\overline{L}$ is not RE then K is not RE either

3. Let take the language $\overline{L}$ and we know $\overline{L}$ is not RE so we know take the function $f(w) = aw$ But if we use f on $\overline{L}$ then all $f(w)$ are not in K, similarly if w is not in $\overline{L}$ then $f(w)$ is in K(As is w is not in $\overline{L}$ it's in L so its of the form av which is in K). So we have $\overline{L} \leq_f \overline{K}$. But as $\overline{L}$ is not RE then $\overline{K}$ is not RE either

1. Can be deduce from the previous two as K is not RE and $\overline{K}$ is not RE either then K is not recursive