1 problem 1

Since L'$\leq_m$L, we have a Polynomial time computable function f, $\forall$x x$\in$L' iff f(x)$\in$L.
Since L$\in$P, we have a det polynomial time algorithm M_L to solve L.
We construct a polynomial time algorithm to solve L'.
for all input x
compute f(x)
If f(x)$\in$L then M_L accepts x, otherwise M_L rejects x.
So if L$\in$P and L'$\leq_m$L, clearly L'$\in$P.

2 problem 2

Since L'$\leq_m$L, we have a Polynomial time computable function f, $\forall$x x$\in$L' iff f(x)$\in$L.
So we know $\forall$x'$\in \overline{L}'$, x'$\notin$L', f(x')$\notin$L, f(x')$\in \overline{L}$
So $\forall$ f(x')$\in \overline{L}$, f(x')$\notin$L, x'$\notin$L', x'$\in \overline{L}'$
So x'$\in \overline{L}'$ iff f(x')$\in \overline{L}$. So $\overline{L}' \leq_m \overline{L}$.

3 problem 3

We assume that recursive language is NP.
This means for any recursive language, there have an polynomial time algorithm accept it.
For example we construct a recursive language L, L=0^i: 0^i is not accepted by Mi in 2^{nk} steps.

But L$\notin$DTIME(2^{nk}). So recursive language is not in NP.

4 problem 4

Let L be a recursive language and L1 = SAT and L2=L1 $\cup$ L. Clearly we know that L1 is NP-complete, and L2 is not in NP.

5 problem 5

First we need show that problem Q is in NP:
Guess S' and check that S' $\subseteq$ S, —S'— $\leq$ k, and S' contains at least one element from each subset in C. Guessing S' takes O(n) time and checking S' takes O(n^2) time, so Q $\in$ NP.
Secondly, we need show problem Q is NP-hard:
VC $\leq_m$ Q: The Vertex Cover problem is NP-complete, so if VC can be transformed into Q in polynomial time, then Q is NP-hard. Transform the Vertex Cover problem to Q as follows:
1. Start with a graph G=(V, E) and value k.
2. Set S=V.
3. For each (v, v') $\in$ E, add the subset v, v' to C.
From S and C, form the set S'. The set S' is a vertex cover for G, where —S'—$\leq$ k. In the original VC problem, the size of the VC is k. If —S'— < k, then $\exists$ S", where S' $\subset$ S" $\subseteq$ S, and —S"—=k. The set S" is formed by setting

S" = S' and adding elements of S until —S"—=k. S" is stille a VC for G. The transformation of VC to Q can be performed in deterministic polynomial time, so VC $\leq_m$ Q. VC is NP-complete, so Q is NP-hard.

6 problem 6

If L $\in$ NP-complete, then L $\in$ NP, and if L $\in$ co-NP-complete, then L $\in$ co-NP.
Firstly, Np $\subseteq$ co-NP: $\forall$L' $\in$ NP, L'$\leq_m$ L, since L $\in$ NP-complete. So we know L' $\in$ co-NP.
Secondly, co-NP $\subseteq$ NP: $\forall$L' $\in$ co-NP, L'$\leq_m$ L. Since L $\in$ NP, L' $\in$ NP.
NP $\subseteq$ co-NP and co-NP $\subseteq$ NP, so we know NP=co-NP.

7 Problem 7

To show that problem R is NP-complete. We just need to show that problem R is NP and Problem R is NP-hard.
Firstly, R $\in$ NP: Guess spanning tree T and check that T is a spanning tree. To check that T is a spanning tree, check that T contains all vertices in G, that the degree of each vertex is no more than k, and that every edge in T is in G. Guessing T takes O(n) time and checking T takes O(n^2) time, so R $\in$ NP. Secondly, HP $\leq_m$ R: THe Hamiltonian Path problem is concerned with finding a path through a graph that visits each vertex once. Transform the HP

problem to Q as follows:
1. Start with a graph G=(V,E).
2. Set k=2.
From G, find a spanning tree T where the degree of each vertex in T is $\leq$ k. THe spanning tree T is a HP for G. The transformation of HP to R can be performed in deterministic polynomial time, so HP $\leq_m$ R. HP is NP-complete, so R is NP-hard.