Please let me know of any typos or questions via WebCT. Depending on how much time I have I may add additional problems/solutions over the weekend.

Problem 1. (10 points) For each of the following languages, determine whether it is:
- R = recursive,
- RE - R = recursively enumerable but not recursive,
- coRE - R = the complement of a recursively enumerable language but not recursive, or
- N = neither recursively enumerable nor the complement of a recursively enumerable language.

\[
\begin{align*}
&M; x : M(x) \neq \lambda \} & \text{RE-R} \\
&M; x : M(x) = \lambda \} & \text{coRE-R} \\
&M; x : M(x) \text{ will not reach cell } 100 \text{ of the tape} \} & \text{R} \\
&M; x : M(x) \text{ halts after } \leq 100 \text{ steps} \} & \text{R} \\
&M ; M(\lambda) = \lambda \} & \text{coRE-R} \\
&M_1; M_2 : L(M_1) = L(M_2) \} & \text{N} \\
&M_1; M_2 : L(M_1) \neq L(M_2) \} & \text{N} \\
&M; x : M(x) \text{ will reach cell } 100 \text{ of the tape} \} & \text{R} \\
&M; x : M(x) \neq \lambda , \text{ but takes more than } 100 \text{ steps} \} & \text{RE-R} \\
&M_1; M_2; M_3 : L(M_1) = L(M_2) \neq L(M_3) \} & \text{N} \\
&M_1; M_2 : \exists x \text{ such that } M_1(x) = \lambda \text{ and } M_2(x) \neq \lambda \} & \text{N} \\
&M_1; M_2 : M_1(\lambda) = \lambda \text{ and } M_2(\lambda) \neq \lambda \} & \text{N} \\
&M ; \forall x M(x) \neq \lambda \} & \text{N} \\
&M ; \exists x M(x) \neq \lambda \} & \text{RE-R} \\
&M; x : M(x) \text{ takes more than } 100 \text{ steps} \} & \text{R} \\
&M ; \exists x \text{ such that the } 3 \text{rd transition of } M(x) \text{ is to the right} \} & \text{R} \\
&M ; M(M) = \lambda \} & \text{coRE-R} \\
&M ; M(M) \neq \lambda \} & \text{RE-R}
\end{align*}
\]

Problem 2. (10 points) Assume the graph on the left shows a network (with capacities) and the graph on the right shows the current flow in the network, construct the derived network and find an augmenting path.
Derived Network and augmenting path.

Problem 3. (10 points) Create a 2-tape Turing Machine which starts with a string of $a$s and $b$s on tape 1 and ends with the same string in reverse on tape 2.

Problem 4. (10 points) Prove that $L = \{M : \exists x \text{ such that } M(x) \neq \perp\}$ is not recursive.

Given an instance of Halting $(M; x)$ construct a Turing Machine $M'$ which on input $y$ which first checks to see if $x = y$. Then

- if $x \neq y$ it moves to a state which loops regardless of what is read from the tape.
- if $x = y$ it simulates $M(x)$.

Note that (by definition) $M'$ will either run forever on all inputs or run forever on all inputs except $x$.

[Yes $\Rightarrow$ Yes] If $M(x)$ halts (ie $(M; x) \in$ HALTING) then $M'$ will halt on $x$ and therefore $M' \in L$.

[No $\Rightarrow$ No] If $M(x)$ does not halt (ie $(M; x) \notin$ HALTING) then $M'$ runs forever on all inputs and therefore $M' \notin L$. 
Problem 5. (10 points) Prove that \( L = \{ M : \forall x \ M(x) \neq \} \) is not recursive.

Given an instance of Halting \((M; x)\) construct a Turing Machine \(M'\) which on input \(y\) which first simulates \(M(x)\) and if this simulation completes moves to the YES state.

Note that (by definition) \(M'\) will either run forever on all inputs or accept all inputs.

- [Yes \(\Rightarrow\) Yes] If \(M(x)\) halts (ie \((M; x) \in \text{HALTING}\)) then \(M'\) accepts \(\Sigma^*\) and therefore \(M' \in L\).
- [No \(\Rightarrow\) No] If \(M(x)\) does not halt (ie \((M; x) \notin \text{HALTING}\)) then \(M'\) runs forever on all inputs and therefore \(M' \notin L\).

Problem 6. (10 points) Prove that \( L = \{ M : \exists x \neq y \text{ such that } M(x) \neq \} \) is not recursive.

Given an instance of Halting \((M; x)\) construct a Turing Machine \(M'\) which on input \(w\) which first checks whether \(w = x\) or \(w = \lambda\) (I’m assuming that \(x \neq \lambda\). If \(x = \lambda\) replace \(\lambda\) with any other string). Then

- if \(w \neq x\) and \(w \neq \lambda\) it moves to a state which loops regardless of what is read from the tape.
- if \(w = x\) or \(w = \lambda\) it returns to the beginning of the tape and simulates \(M(x)\).

Note that (by definition) \(M'\) will either run forever on all inputs or run forever on all inputs except \(x\) and \(\lambda\).

- [Yes \(\Rightarrow\) Yes] If \(M(x)\) halts (ie \((M; x) \in \text{HALTING}\)) then \(M'\) halts on \(x\) and \(\lambda\) and therefore \(M' \in L\).
- [No \(\Rightarrow\) No] If \(M(x)\) does not halt (ie \((M; x) \notin \text{HALTING}\)) then \(M'\) runs forever on all inputs and therefore \(M' \notin L\).

Problem 7. (10 points) Prove that the language below is NP-complete by reducing CLIQUE to L.

\[
L = \{ G; k; \ell : \exists a \text{ set } S \text{ of } k \text{ vertices each of which is connected to at least } \ell \text{ vertices in } S \}
\]

Given an instance of CLIQUE \((G; n)\) construct an instance of \(L\) which is \((G, n, n-1)\). Note that a clique of size \(n\) is (by definition) a set of \(n\) vertices each of which is connected to all \(n-1\) of the others.

Problem 8. (10 points) Prove that the language below is NP-complete by reducing HAMILTONIAN-CYCLE to L.

\[
L = \{ G; k : G \text{ contains a cycle with } k \text{ vertices} \}
\]

Given an instance of HAMILTONIAN-CYCLE \(G\) construct an instance of \(L\) which is \((G; n)\) where \(n\) is the number of vertices in \(G\). Note that a Hamiltonian cycle is (by definition) a cycle which goes through all \(n\) nodes.
Problem 9. (10 points) Prove that the language below is NP-complete by reducing INDEPENDENTSET to $L$.

$$ L = \{ G; k; l : \exists \text{ a set } S \text{ of } k \text{ vertices each of which is connected to at most } \ell \text{ vertices in } S \} $$

Given an instance of INDEPENDENTSET $(G; n)$ construct an instance of $L$ which is $(G; n; 0)$. Note that an independent set of size $n$ is (by definition) a set of $n$ vertices each of which is connected to none of the others.

Problem 10. (10 points) Prove that the language below is NP-complete by reducing CLIQUE to $L$.

$$ L = \{ G_1; G_2; k : \text{both } G_1 \text{ and } G_2 \text{ have a clique of size } k \} $$

Given an instance of CLIQUE $(G; n)$ construct an instance of $L$ $(G; G'; n)$ where $G'$ is the graph with $n$ vertices and $n(n - 1)/2$ edges. Note $G'$ is a clique on $n$ vertices. So $(G; G'; n) \in L$ iff $G$ has a clique of size $n$.

Problem 11. (10 points) Prove that the language below is NP-complete by reducing HAMILTONIAN-CYCLE to $L$.

$$ L = \{ G_1; G_2 : \text{both } G_1 \text{ and } G_2 \text{ have a hamiltonian cycle} \} $$

Given an instance of HAMILTONIAN-CYCLE $G$ construct an instance of $L$ $(G; G')$ where $G'$ is a triangle (3 vertices and 3 edges). It is clear that $G'$ has a hamiltonian cycle. So $(G; G') \in L$ iff $G$ has a hamiltonian cycle.

Problem 12. (10 points) Prove that the language below is NP-complete by reducing INDEPENDENTSET to $L$.

$$ L = \{ (G_1; G_2; k) : \text{both } G_1 \text{ and } G_2 \text{ have independent sets of size } k \} $$

Given an instance of INDEPENDENTSET $(G; n)$ construct an instance of $L$ $(G; G', n)$ where $G'$ has $n$ vertices and no edges. Clearly $G'$ has an independent set of size $n$. So $(G; G', n) \in L$ iff $G$ has an independent set of size $n$. 
Problem 13. (10 points) Show a polynomial time reduction from 3SAT to NodeCover.

Given an instance of 3SAT $\phi$ construct an instance of NodeCover $G, n$ by

1. For each variable $x_i$ create 2 nodes labeled $x_i$ and $\bar{x}_i$ and connect them with an edge.

2. For each clause $c_j = (l_1 \lor l_2 \lor l_3)$ create 3 nodes labeled $l_1, l_2,$ and $l_3$ and connect them with 3 edges.

3. For each node created in step 2 connect it with the node created in step one with the same label.

4. Set $n$ to be the number of variables plus twice the number of clauses.

[Yes $\Rightarrow$ Yes] Assume that $\phi$ is satisfiable and fix a truth assignment which makes $\phi$ true. Let $S = \emptyset$. For each variable $x_i$, if $x_i = T$, add the vertex created in step 1 labeled $x_i$ to $S$ and otherwise add the vertex created in step one labeled $\bar{x}_i$. For each clause, at least one of the three literals is true and let $l_{ij}$ be a literal in clause $j$ which is true. For each clause, add the two vertices which are not labeled $l_{ij}$ which were created in step 2 to the set $S$. Note that the $S$ contains $n$ vertices, that $S$ clearly covers the edges created in steps 1 and 2. $S$ also covers the edges created in step 3 because it connects two vertices with the same label, one created in step 1 and one created in step 2. If that literal is true then the vertex created in step 1 is in $S$ and otherwise the vertex created in step 2 is in $S$. Either way one of the ends is in $S$.

[Yes $\Leftarrow$ Yes] Assume that the $G$ constructed has a set $S$ of $n$ nodes which cover all edges. Since $S$ covers the edges created in step 1, at least one of the nodes labeled $x_i$ or $\bar{x}_i$ created in step 1 must be in $S$. Since $S$ covers the edges created in step 2, at least two of the three vertices created for clause $j$ must be in $S$. Since $S$ has at most $n$ nodes, the “at least” in the previous two sentences can be removed. Note that the selection of $x_i$ or $\bar{x}_i$ can be seen as a truth assignment for $\phi$. This truth assignment satisfies $\phi$ because the unselected node created for each clause must be connected to a selected node created for each variable. So at least one literal in every clause is true.