Problem 1 [10 marks]
The set of real is an uncountable set, i.e. no bijection from the set of reals to the set of natural numbers exists. Refer to Cantor’s proof, using diagonalization, that the set of real numbers in the interval [0,1) is uncountable. (http://en.wikipedia.org/wiki/Cantor's_diagonal_argument)

Why is it not possible to use the same technique to “prove” that no bijection is possible between the set of natural numbers and the rationals in the interval [0,1)

Solution:
We cannot use the same trick and swap all the digits across the diagonal, because although the new number is not in the list, there is no guarantee that it remains a rational.

Problem 2 [15 marks]
Show that the recursively enumerable languages are closed under union, intersection and Kleene-star operation.

Union:
For any two Turing-recognizable language L₁ and L₂, let M₁ and M₂ be the TMs that recognize them. We construct a TM M that recognizes the union of L₁ and L₂.
- For the input w
- Run M₁ and M₂ in parallel.
- If either accept, accept
- If both reject, reject.

If M₁ and M₂ accept w, M accepts w in a finite number of steps. If M₁ and M₂ reject by looping, M also loops.

Intersection:
We construct a TM M’ that recognizes the intersection of L₁ and L₂:
“On input w:
Run $M_1$ on $w$. If it halts and rejects, reject. If it accepts, go to stage 3.
Run $M_2$ on $w$. If it halts and rejects, reject. If it accepts, accept."

If both of $M_1$ and $M_2$ accept $w$, $w$ belongs to the intersection of $L_1$ and $L_2$ and $M'$ will accept $w$ after a finite number of steps else it will reject or loop forever.

**Star:**
We can use a multi tape Turing machine as all multi tapes Turing machines have an equivalent one tape Turing machine.
For any Turing-recognizable language $L$, let $M$ be the TM that recognizes it. We construct a TM $M_1$ that recognizes the star of $L$:
On input $w$, split $w$ into $w_1w_2...w_n$, save each on a tape.
Run $M$ on $w_i$ for $i=1,2,...,n$.
If $M$ accepts each of these string $w_i$, accept.
If reject reached on the possible parts, reject.”
If there is a way to cut $w$ into substrings such $M$ accepts all the substrings, $w$ belongs to the star of $L$ and $M_1$ will accept $w$ after a finite number of steps.

**Problem 3 [10 marks]**
Show that the recursive languages are not closed under complement.

This is a false statement (sadly a typo). Recursive languages are closed under complement and the proof of this is given in the textbook. If you had been following the forum there was a correction posted. So here’s how it goes.

If you proved the given statement, a wrong proof, then u get no credit.
If you proved it for recursively enumerable language you get credit.
If you just wrote a reference or proof, which disproves the given incorrect statement, then u get credit.

**Problem 4 [10 marks]**
For the language $\{a^j b^k c^l \mid j = k \times l, \text{ and } j,k,l \geq 0\}$
a) Give an informal description of a TM for this language
   1. Scan the input from left to right to be sure that it is a member of $a^*b^*c^*$ and reject if it isn’t.
   2. Return the head to the left-hand end of the tape.
   3. Cross off an $a$ and scan to the right until an $a$ occurs. Shuttle between the $b$’s and the $c$’s, crossing off one of each until all $b$’s are gone.
   4. Restore the crossed off $b$’s and repeat stage 3 if there is another $a$ to cross off. If all $a$’s are crossed off, check on whether all $c$’s also are crossed off. If yes, accept; otherwise, reject.

b) Give an informal description of a two stack PDA for this language
Problem 5 [15 marks]
A Non-Deterministic TM (NTM) can compute a problem if and only if a Deterministic TM (DTM) can, i.e. these two models of computations are computably equivalent.

Does this prove that these are complexity equivalent? Why or why not?

This does not prove them to be complexity equivalent. The important point to note here is that a NTM can simulate a DTM and vice versa. Thus they both have the same computational power (proof given in the book).

But the time taken by a DTM to simulate a NTM is exponential. That is some algorithm that runs in polynomial time on a NTM, when simulated by a DTM take exponential time. (actually this is not generalized and has not been proven/disproven for all types of problems, see next problem)

BONUS PROBLEM [guaranteed A+]
Prove or disprove that a NTM is time complexity-equivalent to a DTM, i.e. that time taken by a NTM to compute is of the same order as taken by a DTM.

You are being asked to prove P = NP? problem. Good luck!