Math 136, Study guide for first midterm

The midterm will cover everything we have done up to this point. That is, Chapters 1 and 2, Chapter 3 sections 1-3 and 7, Chapters 4 and 5, and Chapter 6, sections 1 and 6.

You should also consult the lecture notes. These follow the book pretty much, except that I used a different Godel numbering of URM's and various other objects. You should learn the one I used. Also, I paid more attention to which functions and predicates are primitive recursive; that's worth understanding, but it won't be an emphasis of the midterm.

On the midterm, I will ask you to simply repeat some of the basic definitions and theorems. *Exact* re-statements are all that is wanted here. I may also ask you to produce proof outlines, or maybe fragments of proofs, for some of the main theorems. Finally, there will be homework-like problems.

Here is a little more detail on these categories:

I. **Definitions to know.**

(a) P is a URM,

(b) r is a *register configuration*, s is a *computation stage of* P, t is a *computation of* P.

(c) $f_P^{(n)}$, for P a URM.

(d) g is *URM-computable.*

(e) h is *obtained from* f *and* g *by primitive recursion.* f is *primitive recursive*

(f) f is *obtained from* g *by minimalization.* f is *μ-recursive.*

(g) $\phi_e^{(n)}$, $W_e^{(n)}$, $\Phi_U^{(n)}$, $W_U^{(n)}$.

(h) *decidable* predicate, primitive recursive predicate, partially decidable predicate.

(i) *many-one reducibility:* $A \leq_m B$ (cf. page 158 of book).

II. **Theorems to know.**

(a) f is URM-computable iff f is μ-recursive. (Thm. 2.2 on p. 50, but see notes for more detail.)

(b) Church's thesis (not really a theorem, but pretty important!).

(c) Enumeration theorem, Kleene normal form theorem (thm 1.2, page 86, and thm 1.4, page 89, but see notes for more detail). Cor. 1.3, page 88.

(d) Unsolvability of the Halting problem (thm 1.3, p. 102).

(e) $S-m-n$ theorem (thm. 4.1, page 81).

(f) Rice's theorem (thm. 1.7, page 105).

III. **Proof techniques to know.**

Anything that showed up in the homework is fair game, but here are some important ones:

(a) How to design some very simple URMs.

(b) How to calculate the complexity of functions and predicates by building them up from simpler ones.

(c) Diagonalization.

(d) Use of Church's thesis in informal proofs of recursiveness.

(e) Informal "dovetailing" or "time-sharing" algorithms, based on Cor. 1.3, page 88.

(f) Use of $S-m-n$ theorem to prove that certain recursive program-transforming functions exist. (E.g. to prove closure properties are uniform, as in 3.1 on p. 93.)

(g) many-one reduction of one set to another using S-m-n theorem.