C1 **Theory** (25 points)

a. (6.25 points)
Show that

$$L_1 = \{w \in \{a,b\}^* \mid w \text{ contains subword } aab \text{ or } 20\text{th from last symbol of } w \text{ exists } \& = b\} \quad (1)$$

is regular. You may use with*out* proof any standard text book results about regular sets *provided you clearly say which results you are using when.*[1]

b. (6.25 points)
Find a *deterministic* finite automata $\mathcal{M}'$ which accepts the same language (over $\{a,b\}$) as the non-deterministic finite automata $\mathcal{M}$ depicted in table form just below.

δ	a	b
start 1	$\{1,2\}$	$\{1\}$
2	$\{3\}$	$\{3\}$
3	$\{4\}$	$\{4\}$
4	$\{5\}$	$\{5\}$
final 5	$\emptyset$	$\emptyset$.

c. (6.25 points)
Employ an appropriate pumping lemma to show that

$$L_2 = \{a^m b^n \mid m \text{ is a perfect square } \vee \ n \text{ is odd}\} \quad (2)$$

is *not* regular.

d. (6.25 points)
Employ an appropriate pumping lemma to show that

$$L_3 = \{ab^p \mid p \text{ is prime}\} \quad (3)$$

is *not* context free, i.e., is *not* accepted by any push down automaton.

[1] L_1 is *not* a standard text book regular language. (☺)

C2 **Theory** (25 points)

Consider the following finite automaton $\mathcal{M}$ expressed in tabular form.

δ	a	b
start 1	2	4
2	3	4
final 3	3	3
4	2	5
5	2	6
6	6	6.

This $\mathcal{M}$ *is* minimal state (for the accepting task it performs).

Explicitly employ Myhill-Nerode to *prove* this $\mathcal{M}$ is minimal state.

Hint: You may find it useful to draw the state diagram of $\mathcal{M}$.
Find a relevant spanning S by considering how to reach each state of $\mathcal{M}$ from its start state. Show this S can't be reduced in size and still be relevantly spanning. The number of combinations of six things taken two at a time is 15.

C3 **Theory** (25 points)

We consider below functions which have non-negative integer arguments and values. We use Church's lambda-notation to name functions defined by formulas (as is standard in Lisp). For example, $\lambda x.x+1$ denotes the function that maps each non-negative integer x to $x+1$, and $\lambda x, y.x+y$ denotes the function of two non-negative integer arguments mapping them to their sum.

Let N denote the set of non-negative integers.

For each $x, y \in N$, let

$$x \dot{-} y \stackrel{\text{def}}{=} \begin{cases} x-y, & \text{if } x \geq y; \\ 0, & \text{otherwise;} \end{cases} \tag{4}$$

$$p_x \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x = 0; \\ \text{the } x\text{-th prime number}, & \text{otherwise;} \end{cases} \tag{5}$$

and let

$$\langle x, y \rangle \stackrel{\text{def}}{=} 2^x(2y+1) \dot{-} 1. \tag{6}$$

You may assume *without* proof that $\lambda x, y.\langle x, y\rangle$ maps the set of all pairs of non-negative integers 1-1 onto the set of all non-negative integers. Let ℓ and r be the left and right inverses of $\lambda x, y.\langle x, y\rangle$, respectively. Then, we have that, for all x, y, z,

$$\ell(\langle x, y\rangle) = x, r(\langle x, y\rangle) = y, \text{ and } \langle \ell(z), r(z)\rangle = z. \tag{7}$$

You may also assume *without* proof that the functions in the set $\mathcal{S}$ are each primitive recursive, where $\mathcal{S} \stackrel{\text{def}}{=}$

$$\{\lambda x, y.x+y, \lambda x, y.x \times y, \lambda x, y.x \dot{-} y, \lambda x, y.\langle x, y\rangle, \ell, r, \lambda x.p_x, \lambda x.(x > 0)\}.^2 \tag{8}$$

For each $n, x_1, \ldots, x_n \in N$, let

$$[x_1, \ldots, x_n] \stackrel{\text{def}}{=} \Pi_{i=1}^{n} p_i^{x_i}.^3 \tag{9}$$

You may also assume *without* proof that, for each positive $n \in N$, $\lambda x_1, \ldots, x_n.[x_1, \ldots, x_n]$ is primitive recursive. You may further assume *without* proof the existence of primitive recursive functions $\lambda x.\mathrm{Lt}(x)$ and $\lambda x, i.(x)_i$ such that

- $\mathrm{Lt}(0) = \mathrm{Lt}(1) = 0$, and, for each non-negative integer $x > 1$, for each $n \in N$, if $\mathrm{Lt}(x) = n$, then p_n divides x, but *no* prime greater than p_n divides x; and
- for each $x, i \in N$, $(x)_0 = 0$, $(0)_i = 0$, and, if $x, i > 0$, $(x)_i =$ the exponent of p_i in the unique prime power factorization of x.

Definition Suppose $n \in N$. Suppose f has $n+1$ arguments.

f is defined from h by course of values recursion $\stackrel{\text{def}}{\Leftrightarrow}$, for each $x_1, \ldots, x_n, y$,

$$f(x_1, \ldots, x_n, y) = h(y, [f(x_1, \ldots, x_n, 0), f(x_1, \ldots, x_n, 1), \ldots, f(x_1, \ldots, x_n, y-1)], x_1, \ldots, x_n). \tag{10}$$

Finally, you may also assume *without* proof that the class of primitive recursive functions is closed under definition by course of values recursion.

Definition *f is defined from g_1, g_2, and h by intertwined primitive recursion* $\stackrel{\text{def}}{\Leftrightarrow}$

$$(\forall y \in N)[f(0, y) = g_1(y)]; \tag{11}$$

$$(\forall x \in N)[f(x+1, 0) = g_2(x)]; \tag{12}$$

and

$$(\forall x, y \in N)[f(x+1, y+1) = h(x, y, f(x, y+1), f(x+1, y))]. \tag{13}$$

Prove from any or all of the assumptions without proof above *and* the definition of primitive recursive function that, *if* g_1, g_2, and h are primitive recursive and f is defined from g_1, g_2, and h by intertwined primitive recursion, *then* f is also primitive recursive. You need *not* put in all the Base Functions (e.g., projection functions), but give enough detail to show you know what you are doing and don't say false things.

[2] It is to be understood that predicates (such as $\lambda x.(x > 0)$) have value 1 when they are true and 0 when they are false.

[3] Of course, as is standard, an empty product evaluates to 1.

C4 **Theory** (25 points)

Let N denote the set of non-negative integers.

If ψ is a partial function mapping N into N, then $\delta\psi$ denotes the *domain* of ψ, i.e., $\{x \mid (\exists y)[\psi(x) = y]\}$; and $\psi(x)\downarrow$ means $\psi(x)$ is defined. For partial functions ψ, θ mapping N into N; θ *extends* $\psi \overset{\text{def}}{\Leftrightarrow}$

$$(\forall x)[\psi(x)\downarrow \Rightarrow \theta(x)\downarrow = \psi(x)].^4 \tag{14}$$

Fix a standard programming formalism φ for computing all the *one-argument* partial computable functions which map N into N.[5] Code (Gödel) number the φ-programs *onto* N. Let φ_p denote the partial function computed by program (number) p in the φ-system.

Let $\Phi_p(x) \overset{\text{def}}{=}$ the number of steps φ-program p executes on input x if p on x halts and undefined if p on x does not halt. You may assume: $\Phi_p(x)$ defines a *partially* computable function of p, x; $\Phi_p(x)$ is defined exactly when $\varphi_p(x)$ is defined; and

$$\{(p, x, t) \mid \Phi_p(x) \leq t\} \text{ is a computable set.} \tag{15}$$

We write $\Phi_p(x) > t$ to mean *either* $\Phi_p(x)$ is defined to be some number $> t$ *or* $\Phi_p(x)$ is undefined.

You may assume with*out* proof that, in the φ-system, Universality, S-m-n, and the Kleene Recursion Theorem (KRT) hold.

Define

$$f \text{ is } \textit{conservative} \overset{\text{def}}{\Leftrightarrow} [f \text{ is computable: } N \text{ into } N \ \wedge \ (\forall p)[\varphi_{f(p)} \text{ extends } \varphi_p]]. \tag{16}$$

For this problem we lead you through a proof of the following

Theorem (Stuart Kurtz) *Suppose f is a conservative transformation. Suppose ψ is partial computable.*

Then there is an e such that $\varphi_{f(e)} = \psi$.

Suppose, then, the hypotheses of Kurtz' Theorem just above.

a. (5 points)

Prove there is a φ-program p_0 such that

$$\varphi_{p_0} = \psi. \tag{17}$$

b. (10 points)

Argue that there is an e such that (18) just below *by explicitly, informally laying out step by step what algorithmic steps your e takes* on an arbitrary input $x \in N$. **Be sure to say which assumptions above you are using to show which of your steps are algorithmic.**

$$\forall x : \varphi_e(x) = \begin{cases} \psi(x), & \text{if } \Phi_{f(e)}(x) > \Phi_{p_0}(x)\downarrow; \\ 1 + \varphi_{f(e)}(x), & \text{otherwise.} \end{cases} \tag{18}$$

c. (10 points)

Prove next that, for each $x \in N$,

$$\varphi_{f(e)}(x) = \psi(x) = \varphi_e(x) \tag{19}$$

— completing the proof of Kurtz' Theorem.

Hint: For each $x \in N$, consider separately the cases of $x \notin \delta\psi$ and $x \in \delta\psi$.

---

[4] N.B. Clause (14) is a universally quantified *if-then* clause, *not* a universally quantified *iff* clause! In particular, if θ extends ψ, then θ agrees with ψ where ψ is defined and *may* be defined some places ψ isn't.

[5] N.B. We are using a *lower* case Greek phi for this notation. Below we employ the upper case of this Greek letter but for a different purpose.