C1 **Theory** (25 points)

a. (6.25 points)
Show that

$$L_1 = \{w \in \{a,b\}^* \mid w \text{ contains the subword } bba \textit{ or } w \text{ ends in } baa\} \quad (1)$$

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

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

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

c. (6.25 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.

d. (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.

[1] L_1's regularity is *not* a standard text book general result about regular languages. (◡̈)

C2 **Theory** (25 points)

a. (12.5 points)
Explicitly draw a state diagram for a *PDA* which accepts all and only the words in

$$L = \{xayubv \mid x, y, u, v \in \{a, b\}^* \ \wedge \ |x| = |y| \ \wedge \ |u| = |v|\}. \qquad (3)$$

b. (12.5 points)
Let $L = \{a^m b^n a^m b^n \mid m, n > 1\}$. *Explicitly employ Pumping for PDA* to show that L is *not* a Context Free Language, i.e., that it is *not* accepted by any PDA.

C3 **Theory** (25 points)

Let N denote the set of non-negative integers.

Fix an *unknown arbitrary* standard programming formalism for computing all the *one-argument* partial computable functions which map N into N. Fix a code (Gödel) numbering of the programs of this formalism *onto* N. Let φ_p denote the partial function computed by program (number) p in the formalism.

Let $\Phi_p(x) \stackrel{\text{def}}{=}$ 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

$$\{(p, x, t) \mid \Phi_p(x) \leq t\} \text{ is a computable set.} \tag{4}$$

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

Prove by explicit application of KRT the following

Theorem *Suppose f is computable, i.e., partially computable and total. Then there is an e such that*

1. $\varphi_e = f$ *and*
2. $(\forall x \in N)[\Phi_e(x) < \Phi_e(x+1)]$.

Hint for C3: Suppose f is computable. *In*formally apply KRT to get an e *which creates a self-copy* and which, on $x > 0$, uses that self-copy to (try to) compare $\Phi_e(x-1)$ and $\Phi_e(x)$. If, as is *not* desired, $\Phi_e(x-1) \geq \Phi_e(x)$, make sure e does something that, in the case that $\Phi_e(x-1)\downarrow$, i.e., in the case that $\Phi_e(x-1)$ is defined, will yield a contradiction. Otherwise, have e output $f(x)$. That was about $x > 0$. Explicitly make $\varphi_e(0) = f(0)$ — with *no* use of e's self-copy.

When you have the behavior of your e on any input x all worked out *and have justified that your e's use of its self-copy and of its input x is algorithmic*, then argue as follows. Suppose for contradiction that x is the *least* number such that $\varphi_e(x)\uparrow$, i.e., such that $\varphi_e(x)$ is undefined. Argue that, then, $\Phi_e(x)\uparrow$. Argue that $\Phi_e(0)\downarrow$. Show, then, that $[x > 0 \ \wedge \ \Phi_e(x-1) \geq \Phi_e(x)]$. What can you then conclude re $\Phi_e(x-1)$? What can you then conclude re $\varphi_e(x-1)$? Get a contradiction. Argue that, then, φ_e is total. Finish the proof of the theorem.

C4 **Theory** (25 points)

The notation and terminology below is standard from the associated reading list book[2] for this Theory part of the Preliminary Exam *except* that φ is used below in place of that book's Φ.[3]

This question, C4, features four multiple choice problems (about types), **where a short explanation for each of your choices also required. Again: you must also explain each of your choices!**

a. (6.25 points) Which one of the following is a type of φ?

1. Computable function.
2. Snapshot.
3. Infinite partial computable function.
4. Finite partial computable function.
5. $\mathcal{L}$-program.

b. (6.25 points) Which one of the following is a type of $\{2, 10^{10}\}$?

1. $\mathcal{L}$-program.
2. Partial computable function.
3. R.e. set.
4. Computable function.
5. Snapshot.

c. (6.25 points) Which one of the following is a type of K?

1. Non-negative integer.
2. Partial computable function.
3. $\mathcal{L}$-program.
4. Non-r.e. set.
5. R.e. set.

d. (6.25 points) Which one of the following is a type of $\overline{K}$?

1. Computable, {0,1}-valued function.
2. Computable set.
3. R.e. set.
4. Non-r.e. set.
5. Non-negative integer.

[2]This book is: M. Davis, R. Sigal, and E. Weyuker, *Computability, Complexity and Languages: Fundamentals of Theoretical Computer Science*, Second Edition, Academic Press, New York, NY, 1994.

[3]This usage of φ below is also the same as its usage in the CISC 601 course here based on that reading list book.