C1 **Theory** (25 points)

Let N denote the set of non-negative integers. Let I^+ denote the set of positive integers.

Let $\Sigma = \{a, b\}$.

In this problem (C1) we will make reference to the languages L_1, L_2, and L_3 defined just below.

Definition

(i) $L_1 \stackrel{\text{def}}{=} \{a^i b^j \mid i, j \in I^+ \ \wedge \ i \text{ and } j \text{ have no prime factors in common}\}$.

(ii) $L_2 \stackrel{\text{def}}{=} \{a^i b^j c^k \mid i, j, k \in N \ \wedge \ [i \neq j \ \vee \ j \neq k]\}$.

(iii) *Let L be a non-computable language over Σ. Let* $L_3 \stackrel{\text{def}}{=} \{x \mid (\exists y \in L)[xy \in \Sigma^*]\}$

a. (9 points)
Use Myhill-Nerode to show that L_1 is *not* regular.[1]

b. (9 points)
Use Pumping to show that L_2 is not regular.

c. (7 points)
Prove that L_3 *is* regular.

Hint for C1c: Do this by programming in some relevant system(s) and also *document* your "code". *Your documentation should be to help the graders understand what you had in mind.* This problem (C1c) can*not* be done constructively, so your programming should feature **finitely** many non-constructive (i.e., non-algorithmic) choices of relevant parameter values. These choices of *yours* should *not* be confused with non-deterministic choices!

[1] Recall that the crucial equivalence relation for Myhill-Nerode applied to the language L_1 is $\equiv_{L_1}$, where, for words $x, y \in \Sigma^*$,

$$x \equiv_{L_1} y \stackrel{\text{def}}{\Leftrightarrow} (\forall z \in \Sigma^*)[xz \in L_1 \Leftrightarrow yz \in L_1]. \qquad (1)$$

C2 **Theory** (25 points)

N.B. For this problem you may use *without stating how they work or proving them correct* standard text book algorithms in automata theory to:

1. Decide of any (deterministic) finite automata whether or not it accepts the empty language.
2. Decide of any (deterministic) finite automata whether or not it accepts an infinite language.
3. Witness the closure of the regular languages under finitely many applications of the Boolean set operations.

Just be sure to state which text book algorithm you are using when.

Devise, then, an algorithm for deciding of an arbitrary (deterministic) finite automaton $\mathcal{M}$, over the fixed alphabet $\{a, b\}$, whether or not *both* infinitely many words in the language accepted by $\mathcal{M}$ contain the subword *abb and* all the words in this language are of odd length.

C3 **Theory** (25 points)

Let N denote the set of non-negative integers. Fix a standard programming formalism φ for computing all the *one-aryument* partial computable functions which map N into N. Code (Gödel) number the φ-programs *onto* N. Let φ_p denote the partial function computed by program (number) p in the φ-system. You may assume without proof that, in the φ-system, Universality, S-m-n, and the Kleene Recursion Theorem (KRT) hold. *You will not need all these assumptions (for* example, *for this problem (C3) you will not need KRT) — just say which ones you are using where.*

You may use without proof any of the standard text book closure properties for the class of primitive recursive functions *provided you say which closure properties you are using where.* You may also use without proof any standard text book examples of primitive recursive functions, *but you must say what they are.*

You may also cite without proof relevant standard text book results about r.e. sets, *but, again, you must say what they are.*

Let $K \stackrel{\text{def}}{=} \{p \in N \mid \varphi_p(p) \text{ is defined}\}$.

If $A \subseteq N$, then C_A is (by definition) the function from N to $\{0,1\}$ which is 1 on A and 0 on the complement of A.

Let $\langle \cdot, \cdot \rangle$ be a fixed primitive recursive function (with primitive recursive inverses) mapping $N \times N$ one-to-one onto N.

a. (10 points)
Show that, for *each* x, $\{w \mid w \in K \ \wedge \ w < x\}$ is a primitive recursive set.

b. (10 points)
Show there is *no* computable f such that, for all $x \in N$,

$$\varphi_{f(x)} = C_{\{w \mid w \in K \wedge w < x\}}. \tag{2}$$

Hint for C3b: Suppose for contradiction otherwise. Then show K is recursive (i.e., computable) to get a contradiction.

c. (5 points)
Explicitly use C3b just above *and Kleene's S-m-n Theorem* to show that $\{\langle w, x\rangle \in N \times N \mid w \in K \ \wedge \ w < x\}$ is *not* a recursive (i.e., computable) set.

C4 **Theory** (25 points)

Let N denote the set of non-negative integers. Fix a standard programming formalism φ for computing all the *one-argument* partial computable functions which map N into N. Code (Gödel) number the φ-programs *onto* N. Let φ_p denote the partial function computed by program (number) p in the φ-system. You may assume without proof that, in the φ-system, Universality, S-m-n, and the Kleene Recursion Theorem (KRT) hold. *You may not need all these assumptions — just say which ones you are using where.*

Let Φ be a run-time function for φ satisfying the Blum Axioms.

Let $K \stackrel{\text{def}}{=} \{p \in N \mid \varphi_p(p) \text{ is defined}\}$.

If $A \subseteq N$, then C_A is (by definition) the function from N to $\{0,1\}$ which is 1 on A and 0 on the complement of A.

Definition *A function f (mapping N into N) is* limiting-computable $\stackrel{\text{def}}{\Leftrightarrow}$

$$[f \textit{ is total } \wedge\ (\exists \textit{ computable } g)(\forall x)[\lim_{t\to\infty} g(x,t) = f(x)]]. \tag{3}$$

Since the limit in (3) is discrete, (3) is equivalent to

$$[f \text{ is total } \wedge\ (\exists \text{ computable } g)(\forall x)(\overset{\infty}{\forall} t)[g(x,t) = f(x)]], \tag{4}$$

where '$(\overset{\infty}{\forall} t)$' means 'for all but finitely many t'. Intuitively, think of t as a discrete time parameter. Then f is limiting-computable means that f is total and there is a mind-changing algorithm for computing f: on any input $x \in N$, it outputs a succession of guesses $g(x,0), g(x,1), g(x,2), \ldots$ as to the value of $f(x)$ *and*, eventually, these guesses are all correct, i.e., for *some* time t_0, for *all* times $t \geq t_0$, the guess at time t, $g(x,t)$, is, correctly $= f(x)$.

a. (10 points)

Prove that C_K is limiting-computable.

Hint for C4a: For each $t \in N$, let $K^t \stackrel{\text{def}}{=} \{x \mid [x \leq t \ \wedge\ \Phi_x(x) \leq t]\}$.

Let $g(x,t) \stackrel{\text{def}}{=} C_{K^t}(x)$.

b. (15 points)

Employ the very useful hint below to prove the following.

Theorem $(\forall \text{ limiting-computable } H)(\exists e, D \mid D \text{ is finite } \wedge\ \varphi_e = C_D)[H(e) < \text{card}(D)]$.

Hint for C4b: Suppose H is limiting-computable. Let g be a computable function such that $(\forall x)[\lim_{t\to\infty} g(x,t) = H(x)]$.
By convention, let $\max(\emptyset) = 0$.
For a *suitable* e (from KRT) let

$$D = \{x \in N \mid \text{card}(\{w < x \mid \varphi_e(w) = 1\}) \leq g(e,x)\}. \tag{5}$$

Your e should employ *both* self-reference *and* ordinary programming language recursion. Of course *you* must say what *your* e and φ_e are! Prove by mathematical induction that *your* φ_e is *total*. The fact that your e employs recursion will help with this!
Suppose we choose x_0 so large that $(\forall x \geq x_0)[g(e,x) = H(e)]$. Why does x_0 exist?
Let $b \stackrel{\text{def}}{=} \max(\{g(e,x) \mid x \geq 0\})$. Show that $b = \max(\{g(e,x) \mid x < x_0\} \cup \{H(e)\}) < \infty$. Let $D[x] \stackrel{\text{def}}{=} (D \cap \{w \in N \mid w < x\})$. Show that $(\forall x \in D)[\text{card}(D[x]) \leq b]$. Show, then, that $H(e) < \text{card}(D) \leq b+1$. Suppose for contradiction $H(e) \geq \text{card}(D)$. Suppose $x_1 > \max(D)$. Let $x_2 = \max(\{x_0, x_1\})$. Show that $\text{card}(D[x_2]) = \text{card}(D) \leq H(e) = g(e,x_2)$. Show that $x_2 \in D$. Show how to use that to obtain a contradiction.