Problem 1

Problem a

A language is a set of strings over an alphabet. What does it mean that a language is recursive? What does it mean that a language is recursively enumerable? Give the essence of the definitions from Lewis & Papadimitriou's textbook. Give short answers.

———————- Solution:

See Lewis & Papadimitriou's textbook.

Problem b

What does it mean that a function $f : \mathbb{N} \times \ldots \times \mathbb{N} \to \mathbb{N}$ is primitive recursive? Give a short answer.

———————- Solution:

See Lewis & Papadimitriou's textbook.

Problem c

Let $f(x, y) = x+y$ and $g(x, y) = x \times y$. Prove that f and g are primitive recursive functions.

———————- Solution:

We will prove that f and g are primitive recursive functions by providing primitive recursive definitions of f and g, that is, by defining the functions from the primitive recursive basic functions (**succ**,$\mathbf{id}_{k,j}$, $\mathbf{zero}_k$) by a number of of successive applications of *composition* and *(primitive) recursion.*

- We define h by composition: $h(x_1, x_2, x_3) = \mathbf{succ}(\mathbf{id}_{3,3}(x_1, x_2, x_3))$.
- We define f by recursion: $f(x, 0) = \mathbf{id}_{1,1}(x)$ and $f(x, y + 1) = h(x, y, f(x, y))$.
- We define h' by composition: $h'(x_1, x_2, x_3) = f(\mathbf{id}_{3,1}(x_1, x_2, x_3), \mathbf{id}_{3,3}(x_1, x_2, x_3))$.
- We define g by recursion: $g(x, 0) = \mathbf{zero}_1(x)$ and $g(x, y + 1) = h'(x, y, f(x, y))$.

The solution given above is very detailed and elaborated. The following solution is satisfactory and will give full score:

- $f(x, 0) = x$ and $f(x, y + 1) = \mathbf{succ}(f(x, y))$.
- $g(x, 0) = 0$ and $f(x, y + 1) = f(x, g(x, y))$.

Problem 2

Let S be a unary function symbol, and let a and 0 be constant symbols. Let $\mathcal{L}$ be the first-order language $\{a, 0, S\}$, and let Σ_0 be the $\mathcal{L}$-theory consisting of the following three non-logical axioms:

(ax1) $\forall x[0 \neq Sx]$

(ax2) $\forall xy[Sx = Sy \ \rightarrow \ x = y]$

(ax3) $a = Sa$.

Problem a

Give two $\mathcal{L}$-structures $\mathfrak{A}$ and $\mathfrak{B}$ such that $\mathfrak{A} \models \Sigma_0$ and $\mathfrak{B} \not\models \Sigma_0$.

———————- Solution:

Let $A = \mathbb{N} \cup \{\alpha\}$. Let $0^{\mathfrak{A}} = 0$ and $a^{\mathfrak{A}} = \alpha$ and

$$S^{\mathfrak{A}}(x) \;=\; \begin{cases} x+1 & \text{if } x \in \mathbb{N} \\ x & \text{otherwise.} \end{cases}$$

Let $B = \mathbb{N}$. Let $0^{\mathfrak{B}} = a^{\mathfrak{B}} = 0$ and $S^{\mathfrak{B}}(x) = x + 1$.

———————

> **Lemma 1.** Let $t_1, \ldots, t_n$ be substitutable for respectively $x_1, \ldots, x_n$ in the $\mathcal{L}$-formula ϕ. Let $\phi^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$ denote ϕ where $x_1, \ldots, x_n$ are replaced by respectively $t_1, \ldots, t_n$. Then, $\forall x_1, \ldots, x_n \phi \vdash \phi^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$.

Problem b

Prove Lemma 1.

———————- Solution:

We prove the lemma by induction on n.

Case $n = 0$: OK, since $\phi \vdash \phi$.

Induction step:

1.	$\forall x_1, \ldots, x_n \phi \vdash \forall x_n \phi^{x_1,\ldots,x_{n-1}}_{t_1,\ldots,t_{n-1}}$		by ind. hyp.
2.	$\forall x_1, \ldots, x_n \phi \vdash \forall x_n \phi^{x_1,\ldots,x_{n-1}}_{t_1,\ldots,t_{n-1}}$	$\rightarrow \; \phi^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$	Q1
3.	$\forall x_1, \ldots, x_n \phi \vdash \phi^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$		1,2,PC

Problem c

Prove that $\Sigma_0 \vdash 0 \neq a$ by providing a derivation. The proof should not refer to any lemmas or theorems from the textbook, but you can refer to Lemma 1.

———————- Solution:

We give a Σ_0-derivation of $0 \neq a$.

1.	$0 \neq Sa$	ax1, Lemma 1.
2.	$a = Sa$	ax3
3.	$0 = 0$	by E1
4.	$0 = 0 \wedge a = Sa \ \rightarrow \ (0 = a \ \rightarrow \ 0 = Sa)$	by E3
5.	$(0 = a \ \rightarrow \ 0 = Sa) \ \rightarrow \ (0 \neq Sa \ \rightarrow \ 0 \neq a)$	PC (tautology)
6.	$0 = 0 \wedge a = Sa \ \rightarrow \ (0 \neq Sa \ \rightarrow \ 0 \neq a)$	PC,4,5
7.	$0 \neq Sa \ \rightarrow \ 0 \neq a$	PC,2,3,6
8.	$0 \neq a$	PC,1,7

The derivation given above is detailed (yet it is not a full derivation). A derivation like e.g. the following one, will give full score:

1.	$0 \neq Sa$	ax1, Lemma 1.
2.	$a = Sa$	ax3
3.	$a = Sa \ \rightarrow \ (0 \neq Sa \ \rightarrow \ 0 \neq a)$	by E3
4.	$0 \neq a$	PC,1,2,3

Let $\overline{0} = 0$ and $\overline{n+1} = S\overline{n}$.

Problem d

Prove that $\Sigma_0 \vdash a \neq \overline{n}$ for any $n \in \mathbb{N}$. Use induction on n.

---------- Solution:

Induction start: See the solution of Problem **c**.

Induction step:

1.	$a \neq \overline{n}$	ind. hyp.
2.	$a = Sa$	ax3
3.	$Sa = S\overline{n} \ \rightarrow \ a = \overline{n}$	ax2, Lemma 1
4.	$Sa \neq S\overline{n}$	PC,1,3
5.	$a = Sa \ \rightarrow \ (Sa \neq S\overline{n} \ \rightarrow \ a \neq S\overline{n})$	by E3
6.	$a \neq S\overline{n}$	PC,2,4,5
7.	$a \neq \overline{n+1}$	$\equiv$, 6

We extend $\mathcal{L}$ with a unary function symbol f and a binary relation symbol $\sqsubseteq$. Let Σ_1 be Σ_0 extended with the non-logical axioms

(ax4) $x \sqsubseteq y \leftrightarrow x = a \vee x = y$

(ax5) $x \sqsubseteq y \rightarrow f(x) \sqsubseteq f(y)$.

Lemma 2. For any $\mathcal{L}$-formula ϕ, any $\mathcal{L}$-sentence η and any variable x, we have

1. $\vdash \phi$ if and only if $\vdash \forall x\phi$
2. $\eta \vdash \phi$ if and only if $\vdash \eta \rightarrow \phi$.

Problem e

Prove that

$$\Sigma_1 \vdash f(a) \neq a \rightarrow \forall x[f(a) = f(x)]$$

by providing a derivation. You might find Lemma 1 and Lemma 2 helpful.

———————- Solution:

Let $\Gamma = \Sigma_1 \cup \{f(a) \neq a\}$. We give a Γ-derivation of $\forall x[f(a) = f(x)]$. By Lemma 2, we have $\Sigma_1 \vdash f(a) \neq a \rightarrow \forall x[f(a) = f(x)]$.

1.	$f(a) \neq a$	Γ
2.	$a \sqsubseteq x \leftrightarrow a = a \vee a = x$	ax4, Lemma 1 and 2
3.	$a = a$	by E1
4.	$a \sqsubseteq x$	PC,2,3
5.	$a \sqsubseteq x \rightarrow f(a) \sqsubseteq f(x)$	ax5, Lem. 1 and Lem. 2.1
6.	$f(a) \sqsubseteq f(x)$	PC,4,5
7.	$f(a) \sqsubseteq f(x) \leftrightarrow f(a) = a \vee f(a) = f(x)$	ax4, Lemma 1 and 2
8.	$f(a) = f(x)$	PC,1,6,7
9.	$\forall x[f(a) = f(x)]$	8, Lemma 2

Problem f

Prove that there exists a closed quantifier-free formula ξ such that $\Sigma_1 \nvdash \xi$ and $\Sigma_1 \nvdash \neg\xi$. (A formula is closed when there are no free variables in the formula.)

———————- Solution: We give two $\mathcal{L}$-structures $\mathfrak{A}$ and $\mathfrak{B}$ such that

1. $\mathfrak{A} \models \Sigma_1$ and $\mathfrak{B} \models \Sigma_1$
2. $\mathfrak{A} \models \neg f(a) = 0$ and $\mathfrak{B} \models f(a) = 0$.

It follows by the Soundness Theorem for first-order logic that $\Sigma_1 \nvdash f(a) = 0$ and $\Sigma_1 \nvdash \neg f(a) = 0$.

Let $A = B = \mathbb{N} \cup \{\alpha\}$. Let $0^{\mathfrak{A}} = 0^{\mathfrak{B}} = 0$ and $a^{\mathfrak{A}} = a^{\mathfrak{B}} = \alpha$ and

$$S^{\mathfrak{A}}(x) = S^{\mathfrak{B}}(x) = \begin{cases} x+1 & \text{if } x \in \mathbb{N} \\ x & \text{otherwise} \end{cases}$$

and

$$\sqsubseteq^{\mathfrak{A}} \;=\; \sqsubseteq^{\mathfrak{B}} \;=\; \{\, \langle x, y\rangle \mid x = \alpha \vee x = y \,\} \,.$$

Let $f^{\mathfrak{A}}(x) = x$ (the identity function) and $f^{\mathfrak{B}}(x) = 0$ (the constant function 0).

Let Σ_2 be Σ_1 extended with the non-logical axiom $f(a) = 0$.

Problem g

Prove the following assertion: For any closed $\mathcal{L}$-term t there exists $n \in \mathbb{N}$ such that

$$\Sigma_2 \vdash t = \overline{n} \;\vee\; t = a \,.$$

---- Solution:

> **Claim 1.** We have $\Sigma_2 \vdash f(t) = 0$ for any term t.

We prove the claim. It is trivial that $\Sigma_2 \vdash f(a) = 0$. Furthermore, we have $\Sigma_2 \vdash 0 \neq a$ and $\Sigma_2 \vdash f(a) \neq a \;\rightarrow\; \forall x[f(a) = f(x)]$ by Problem **c** and **e**. Hence, by the Soundness Theorem for first-order logic, $\Sigma_2 \models \forall x[f(x) = 0]$. The claim follows by the Completeness Theorem for first-order logic. This completes the proof of Claim 1.

> **Claim 2.** For any closed term t, we have either (1) $\Sigma_2 \vdash t = \overline{n}$ for some $n \in \mathbb{N}$ or (2) $\Sigma_2 \vdash t = a$.

We prove Claim 2 by induction over the structure of the term t. We will apply the Completeness Theorem for first-order logic without referring explicitly to this theorem.

The cases $t \equiv a$ and $t \equiv 0$ hold by E1. Assume $t \equiv St'$. By the ind. hyp. we have either (1) $\Sigma_2 \vdash t' = \overline{n}$ for some $n \in \mathbb{N}$ or (2) $\Sigma_2 \vdash t' = a$. In case (1), we have $\Sigma_2 \vdash t = \overline{n+1}$ by E2; in case (2), we have $\Sigma_2 \vdash t = a$ by the non-logical axiom ax3. Assume $t \equiv f(t')$. We have $\Sigma_2 \vdash t = 0$ by Claim 1. We do not need the induction hypothesis in this case. This completes the proof of Claim 2.

Our proof calculus embodies the inference rule PC. Hence, it follows straightforwardly from Claim 2 that for any closed term t there exists $n \in \mathbb{N}$ such that $\Sigma_2 \vdash t = \overline{n} \;\vee\; t = a$

Let

$$\Gamma = \{\phi \mid \phi \text{ is a closed quantifier-free formula and } \Sigma_2 \vdash \phi\} \,.$$

Problem h

The set Γ can obviously be regarded as a language, that is, as a set of strings over an alphabet. Prove that Γ is a recursive language.

———————————- Solution:

Claim 3. For any closed terms t_1 and t_2, we have $\Sigma_2 \vdash t_1 = t_2$ or $\Sigma_2 \vdash t_1 \neq t_2$

Claim 3 follows from Claim 2 and the following facts

- $\vdash x = x$
- $\vdash x = y \wedge y = z \rightarrow x = z$
- $\vdash x = y \rightarrow y = x$
- $\Sigma_2 \vdash \overline{m} \neq \overline{n}$ if $m \neq n$
- $\Sigma_2 \vdash a \neq \overline{n}$ for any n (Problem **d**)

Claim 4. For any closed quantifier-free formula ϕ, we have $\Sigma_2 \vdash \phi$ or $\Sigma_2 \vdash \neg\phi$.

Prove Claim 4 by induction on the structure of ϕ. When ϕ is an atomic formula, the claim holds by Claim 3. The cases when $\phi \equiv \neg\psi$ and $\phi \equiv \psi \vee \xi$ are straightforward.

The set Γ is r.e., that is, a Turing machine can enumerate the strings in Γ. Let $\gamma_0, \gamma_1, \gamma_2, \ldots$ be such an enumeration. By Claim 4, we know that for any closed quantifier-free formula ϕ there exists $i \in \mathbb{N}$ such that $\gamma_i \equiv \phi$ or $\gamma_i \equiv \neg\phi$.

Hence, a Turing machine M can decide the language Γ. The Turing machine M works as follows: First the Turing machine decides if its input w is a (syntactically correct) closed quantifier-free formula; if not M rejects w. If w is a closed quantifier-free formula, then M proceeds by generating the strings in the list $\gamma_0, \gamma_1, \gamma_2, \ldots$ one by one. If w shows up, then M accepts w; if $\neg w$ shows up, then M rejects w. The Turing machine will either accepts or reject w as either w or $\neg w$ will show up. Thus, Γ is a recursive language.