exercise 1

(a) We show by induction that for any \( k \in \mathbb{N}^+ \), \( c_k : \mathbb{N}^k \rightarrow \mathbb{N} \) is primitive recursive:

**Base (B):** \( c_1(x) = p_1^x \) is primitive recursive.

**Inductive hypothesis (H):** for some \( k \in \mathbb{N}^+ \), \( c_k \) is primitive recursive.

**Inductive step (S):** if \( c_k \) is primitive recursive, then \( c_{k+1} \) is primitive recursive.

Proof of (B): For any \( y, x \in \mathbb{N} \), \( y^x \) is primitive recursive (Cutland 2.4.15). Hence, for any \( x \in \mathbb{N} \), \( p_1^x \) is primitive recursive, by substitution of \( p_1 \) for \( y \) in \( y^x \).

Proof of (S): For any \( i \in \mathbb{N} \), \( p_i \) is primitive recursive (Cutland 2.4.5), and for any \( y, x_1 \in \mathbb{N} \), \( y^{x_1} \) is primitive recursive (Cutland 2.4.15). Hence, for any \( x_i, i \in \mathbb{N} \), \( p_i^{x_i} \) is primitive recursive, by substitution of \( p_i \) for \( y \) in \( y^{x_i} \). Furthermore, for any \( x, y \in \mathbb{N} \), \( xy \) is primitive recursive (Cutland 2.4.5). Hence for any \( k, x_k \in \mathbb{N} \), if \( c_k \) is primitive recursive, \( (c_k)p_k^{x+k_i} \) is primitive recursive, by substitution. Since \( (c_k)p_k^{x+k_i} = c_{k+1}, c_{k+1} \) is primitive recursive.

We conclude by induction that for any \( k \in \mathbb{N}^+ \), \( c_k \) is primitive recursive. \( \square \)

(b) For all \( k \in \mathbb{N} \), \( c_k \) is a bijection between \( \mathbb{N}^k \) and \( \mathbb{N} \). If we write \( c_k(x_1, \ldots, x_k) \) as \( \alpha \), then we define \( c_k^{-1}(\alpha) = (((\alpha)_1, \ldots, (\alpha)_k) \).

We then define the function \( g : \bigcup_{k>0} \mathbb{N}^k \rightarrow \mathbb{N} \) as:

\[
g(x_1, \ldots, x_k) = \pi(k, c_k(x_1, \ldots, x_k)) - 1
\]

Obviously, \( g \) is bijective, with \( g^{-1}(x) = ((\pi_2(x + 1))_1, \ldots, (\pi_2(x + 1))_{\pi_1(x + 1)}) \).

We give a procedure for computing \( g(x_1, \ldots, x_k) \):

1. count the number \( k \) of elements in the sequence \( (x_1, \ldots, x_k) \)
2. compute \( c_k(x_1, \ldots, x_k) \)
3. compute \( \pi(k, c_k(x_1, \ldots, x_k)) - 1 \)

Step 1 satisfies our intuitive notion of computability since the sequence \( (x_1, \ldots, x_k) \) is finite. The operation performed in step 2 is also computable since for all \( k \in \mathbb{N} \), \( c_k \) is primitive recursive. Then, since \( \pi \) is also primitive recursive, and since \( k \) and \( c_k(x_1, \ldots, x_k) \) have been computed in previous steps of the procedure, the operation performed in step 3 is computable. Therefore by the Church thesis, \( g \) is computable.

This question proves the closure of primitive recursive functions under course-of-values recursion (also known as definition by strong induction).

Write \( \bar{x} = x_1, \ldots, x_k \). Let \( f(\bar{x}) \) and \( g(\bar{x}, y, p) \) be primitive recursive functions. Define the function

\[
h(\bar{x}, y) = \begin{cases} f(\bar{x}) & \text{if } y = 0 \\ g(\bar{x}, z, c_y(h(\bar{x}, 0), \ldots, h(\bar{x}, z))) & \text{if } y = z + 1 \end{cases}
\]

Write the function
You will show that $h$ is primitive recursive.

(a) Consider the auxiliary function

$$\tilde{h}(\bar{x}, y) = c_{y+1}(h(\bar{x}, 0), \ldots, h(\bar{x}, y))$$

Prove that $\tilde{h}$ is primitive recursive.

Proof.

When $y = 0$, $\tilde{h}(\bar{x}, y) = c_{1}(f(\bar{x}))$.

When $y = z + 1$, $\tilde{h}(\bar{x}, y) = c_{y+1}(h(\bar{x}, 0), \ldots, h(\bar{x}, y))$

$$= p_{y+1}^{h(\bar{x}, y)} \cdot c_{y}(h(\bar{x}, 0), \ldots, h(\bar{x}, z))$$

$$= p_{y+1}^{\tilde{h}(\bar{x}, z, c_{y}(h(\bar{x}, 0), \ldots, h(\bar{x}, z)))} \cdot c_{y}(h(\bar{x}, 0), \ldots, h(\bar{x}, z))$$

Since $c_{y}(h(\bar{x}, 0), \ldots, h(\bar{x}, z)) = \tilde{h}(\bar{x}, z),\tilde{h}(\bar{x}, y) = \begin{cases} c_{1}(f(\bar{x})) & \text{if } y = 0 \\ p_{y+1}^{\tilde{h}(\bar{x}, z, c_{y}(h(\bar{x}, 0), \ldots, h(\bar{x}, z)))} \cdot \tilde{h}(\bar{x}, z) & \text{if } y = z + 1 \end{cases}$

Given $f(\bar{x})$ and $g(\bar{x}, y, p)$ are primitive recursive, $f'(\bar{x}) = 2f(\bar{x})$ and $g'(\bar{x}, z, p) = p_{z+2}^{g(\bar{x}, z, p)} \cdot p$ are clearly primitive recursive. Now $\tilde{h}$ can be defined by primitive recursion as

$$\tilde{h}(\bar{x}, y) = \begin{cases} f'(\bar{x}) & \text{if } y = 0 \\ g'(\bar{x}, z, \tilde{h}(\bar{x}, z)) & \text{if } y = z + 1 \end{cases}$$

Therefore $\tilde{h}$ is primitive recursive.

(b) Use $\tilde{h}$ to prove that $h$ is primitive recursive.

Proof. $h(\bar{x}, y)$ can be defined as:

$$h(\bar{x}, y) = \begin{cases} f(\bar{x}) & \text{if } y = 0 \\ g(\bar{x}, z, \tilde{h}(\bar{x}, z)) & \text{if } y = z + 1 \end{cases} = \left(\tilde{h}(\bar{x}, y)\right)_{y+1}$$

Thus, $h$ is obtained from primitive recursive functions by composition.
Frege's idea. Let \( A \) be a set of all \( n \) s.t. \( f(n) \neq f(n) \).

Let's exercise CTS:

Create a table with 2 columns: \( x \), \( f(x) \)

We have an effective method of filling in this table up to arbitrary \( x_{\max} \) as follows:

1. Write \( \frac{1}{3} \) in the left column and compute
2. Write the values of \( f(1) \), \( f(2) \), \( f(3) \), \( f(x_{\max}) \)

In the right column (possible since \( f \) was defined).

We now define an inductively couple operation \( \times_A \) on \( X_A \) which, by CTS, will suffice for a general recursive \( X_A \).

The operation is defined as follows:

Given \( n \), generate the table described above
up to row \( \{ n, f(n) \} \) (i.e. \( n = x_{\max} \)).

Next, go up a row at a time, starting with \( (n-1) \). Now

If at any row the value in the right column \( f(n) \) up already

computed, \( \exists m < n, f(m) = f(n) \), so

return 0

If however we reach the row with \( 1 \) in the left column and \( f(1) = f(1) \) then \( \# m \in X = f(1) \), return 0.

Since \( X_A \) is equivalent to a general recursive \( f_A(x) \), it returns 1 if \( x \in A \) and 0 if \( x \notin A \), \( A \) is a recursive set.
Problem 4:

a) We use the bijection \( \Phi \) from the proof of Theorem 1.3 in Chapter 4.

\[
\Phi(T(3,4)) = 4 \cdot \Pi(2,3) + 2 = 4 \cdot (2^2(2 \cdot 3 + 1) - 1) + 2 = 16 \cdot 7 - 2 = 110 = a_1.
\]

\[
\Phi(S(3)) = 4(3-1) + 1 = 4 \cdot 2 + 1 = 9 = a_2.
\]

\[
\Phi(Z(1)) = 4(1-1) = 0 = a_3.
\]

Then,

\[
\gamma(P) = 2^{a_1} + 2^{a_1+a_2+1} + 2^{a_1+a_2+a_3+2} - 1 = 2^{110} + 2^{120} + 2^{121} - 1 \approx 3.989 \times 10^{36}
\]

\[
\gamma(P) \approx 3.989 \times 10^{36}
\]
b) We wish to find the program with code number 100.

\[ 100 = 2^0 + 2^2 + 2^5 + 2^6 - 1 = \]
\[ = 2 + 2^{a_1 + a_2 + 1} + 2^{a_1 + a_2 + a_3 + 2} + 2^{a_1 + a_2 + a_3 + a_4 + 3} - 1 = \]
\[ = 2 + 2^{0 + 1 + 1} + 2^{0 + 1 + 2 + 2} + 2^{0 + 1 + 2 + 0 + 3} - 1. \]

Therefore, we have that

\[ a_1 = 0; \]
\[ a_2 = 1; \]
\[ a_3 = 2; \]
\[ a_4 = 0; \]

Then,

\[ a_1 \ast = B(I_4) = 4(1 - 1) = 0; \]

Hence, \( I_4 \in \mathbb{Z}(1) \).
Therefore, $I_2 : S(1)$.

Similarly,

\[ a_3 = \beta(I_3) = 4\pi (1-1, 1-1) + 2 = 2; \]

This indicates that $I_3 : T(1,1)$.

Finally,

\[ a_4 = \beta(I_4) = 4(1-1) = 0; \]

Therefore, $I_4 : Z(1)$.

The complete program $P$ is

\[
\begin{align*}
I_1 & : Z(1) \\
I_2 & : S(1) \\
I_3 & : T(1,1) \\
I_4 & : Z(1)
\end{align*}
\]

Given a computable function $f$, let $P$ be a URM program that computes $f$, and for

\[
\begin{align*}
\text{for } n \in \mathbb{N} \geq 0, \\
\text{let } P_n \text{ be the program formed by appending the instruction } T(n) \text{ to } P.
\end{align*}
\]

Then $P_2, P_3, P_4, \ldots$ are distinct programs (with distinct indices) that all compute $f$.

This completes the proof.