Proof that there exists a function that it is not \( f_a \).

- Explain what is a code, \( C \).
- Code: You start with an expression \( f_a(0,1,\ldots) \).
  - Use later to translate into ASCII
  - \( \varepsilon \to \backslash \varnothing \), \( \forall \to \backslash \forall \), \( \circ \to \backslash \circ \), \( \circ \to \backslash \circ \).
  - You use ASCII code to transform this expression into 0s and 1s.
  - Append a in front of this sequence of 0s and 1s.
  - We will interpret the resulting sequence as a natural number. This number is called a code of a \( f_a \) function.

\[ \text{Proof:} \]

\[ \text{Let's define a function} \]

- Lemma: There exists an algorithm \( U \) that
  - given a natural number \( C \), checks whether
    - \( C \) is a code of a \( f_a \) function, and
  - if yes, returns the executable file computing the value of this \( f_a \) function.

\[ \text{This file will be denoted } f_C. \]
Let's define a function
\[ f(c) = \begin{cases} \frac{c}{c} + 1 & \text{if } c \text{ is a code of a P.R. function} \\ \emptyset & \text{otherwise} \end{cases} \]
This can be 2 or 3, etc.
We add for the proof.

We will prove that \( f \) is computable and that \( f \) is not P.R.

To prove that \( f \) is computable, we show how to compute it.

First, we use alg. \( V \) to check whether \( c \) is a code of a P.R. function if not we return \( \emptyset \).
If yes, we use alg. \( V \) to produce \( f(c) \)
we run \( f(c) \) and we use \( c \) as the input.
Finally, we add one to the result \( f(c) \)
so we get \( f(c) + 1 \)

So \( f \) is computable. Now prove that \( f \) is not P.R.
We prove by contradiction.
Let's assume that \( f \) is P.R. and let's show that this assumption leads to a contradiction.
Since \( f \) is P.R. it has a code, let's denote this code as \( C_f \)

So, by apply \( V \) to \( C_f \), we get an executable file \( f_{C_f} \) that computes the function \( f(c) \).
This means that for every input \( n \)
\[ f_{C_f}(n) = f(n) \]

This is true for every \( n \).
In particular, it is true for \( n = c_0 \)
\[ f_{C_f}(c_0) = f(c_0) \]

Let's go back to the definition of function \( f \),
What is \( f(c_0) \)?
Is \( c_0 \) a code of a P.R. function? Yes, see here.
What is F(x)? Is it a code of a P.R. function? Yes, see here.

Since F(x) is a code, 
\[ F(\omega) = e_{\omega}(\omega) + 1 \]
\[ \omega = 1 \]
So F is not P.R.

All for loops can be computed with while loops, but not the other way around.

```c
int sum=0;
for(int i = 1; i <= b; i++)
  sum=sum+i;
```

The value of sum depends on the parameter a and on the moment of time m.

1. \( \text{sum}(a,0) = a \)
2. The value of sum depends on the previous moment of time \( t \):
   \( \text{sum}(a,m+1) = \text{sum}(a,m) + 1 \)

- Factorial
```c
int fact=1;
for(int i = 1; i <= a; i++)
  fact=fact \* i;
```

```
f(n_1, \ldots, n_k, 0) = g(n_1, \ldots, n_k)
f(n_1, \ldots, n_k, m+1) = h(n_1, \ldots, n_k, m, f(n_1, \ldots, n_k, m))$
```
```
int f = g(n_1, \ldots, n_k);
for(int i = 1; i <= a; i++)
  f = h(n_1, \ldots, n_k, i-1, f);
```

Composition:
```
\( f \circ g \) = \text{P.R}(D, \emptyset, \text{mult}(\mathbb{N}_2, \mathbb{N}_2))\]
\[ F(0) = g(0) = 1 \]
\[ F(m+1) = h_{m, F(m)} = F(m) \times (m+1) \]

\[
\begin{align*}
& F = 1; \\
& \text{for } \text{int } i = 0; i < n; i++ \\
& \quad F = F \times (i-1) + 1; \\
& F = F + 1; \\
& f = F + 1;
\end{align*}
\]

\[ f(a, b) = \begin{cases} 
2 & \text{if } a \in \{0, 1\} \text{ and } b \in \{0, 1\} \\
\text{undefined} & \text{else}
\end{cases} \]

\[
\begin{align*}
& 0 \quad \text{if } a = \emptyset \land b = \emptyset \\
& 0 \quad \text{if } a = \emptyset \land b = 1 \\
& 0 \quad \text{if } a \in \{0, 1\} \land b = \emptyset \\
& 1 \quad \text{if } a \in \{0, 1\} \land b = 1 \\
& \text{undefined} \quad \text{otherwise}
\end{align*}
\]

\[ \mu m \{ (a = \emptyset \land b = \emptyset \land m = \emptyset) \} \cup (\ldots) \]