Sample Questions

Question 1:
Write a program \( P \) that computes \( \Phi^1_P(1)(r) = r - 2 \) using no macros. 

Notice that we only consider natural numbers! This means that, for example, the expression \((1-2)\) is undefined. Any program computing such an expression must never halt.

Sample solution:

\[
\begin{align*}
X & \leftarrow X - 1 \\
\text{IF } X \neq 0 \text{ GOTO } A_2 \\
A_1 & \\
Z & \leftarrow Z + 1 \\
\text{IF } Z \neq 0 \text{ GOTO } A_1 \\
A_2 & \\
X & \leftarrow X - 1 \\
Y & \leftarrow Y + 1 \\
\text{IF } X \neq 0 \text{ GOTO } A_2 \\
Y & \leftarrow Y - 1 \\
\end{align*}
\]

Sample Questions

Question 2:

a) Write a program \( S \) that computes \( \Phi^1_S(1)(r) = r - 4 \). Use a macro based on program \( P \) of the form \( W \leftarrow p(V) \) (which has the effect \( W \leftarrow V - 2 \)).

Sample solution:

\[
\begin{align*}
Y & \leftarrow p(X) \\
Y & \leftarrow p(Y) \\
Z_2 & \leftarrow 0 \quad \text{ // expansion of } Y \leftarrow p(X) \\
& \quad \text{ with } m = 2 \\
Z_3 & \leftarrow X \\
Z_4 & \leftarrow 0 \\
Z_4 & \leftarrow Z_1 - 1 \\
\text{IF } Z_4 < 0 \text{ GOTO } A_1 \\
A_1 & \\
Z_5 & \leftarrow Z_1 - 1 \\
\text{IF } Z_5 < 0 \text{ GOTO } A_1 \\
A_1 & \\
Z_6 & \leftarrow Z_1 - 1 \\
\text{IF } Z_6 < 0 \text{ GOTO } A_1 \\
A_1 & \\
Y & \leftarrow Z_1 \\
Z_6 & \leftarrow Z_1 - 1 \\
\text{IF } Z_6 < 0 \text{ GOTO } A_1 \\
Y & \leftarrow Z_1 - 1 \\
Z_7 & \leftarrow Z_1 - 1 \\
\text{IF } Z_7 < 0 \text{ GOTO } A_1 \\
A_1 & \\
Z_8 & \leftarrow Z_1 - 1 \\
\text{IF } Z_8 < 0 \text{ GOTO } A_1 \\
A_1 & \\
Z_9 & \leftarrow Z_1 - 1 \\
\text{IF } Z_9 < 0 \text{ GOTO } A_1 \\
A_1 & \\
\end{align*}
\]

Macros

Let \( P(x_1, \ldots , x_n) \) be a computable predicate. Then we expand the macro \( \text{IF } P(V_1, \ldots , V_n) \text{ GOTO } L \) to

\[
\begin{align*}
Z & \leftarrow P(V_1, \ldots , V_n) \\
\text{IF } Z = 0 \text{ GOTO } L \\
\end{align*}
\]

As usual, the variable \( Z \) has to be chosen to create no conflicts with the main program.
(Partially) Computable Functions
By introducing macros, we have seen that it is possible to compute complex functions with our very simple programming language $L$. Notice that macros do not change the specification of the language, but they just simplify writing down programs. As you know, we can always replace macros with actual code. So what are the limitations of the language $L$? In order to find out, we need to do some mathematics.