Sample Questions

Question 1:
Write a program $P$ that computes $\Phi_P(1)(r) = r-2$ 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:

```
IF X \neq 0 GOTO B
[A] Z \leftarrow Z+1
IF Z = 0 GOTO C
[B] X \leftarrow X-1
IF X = 0 GOTO D
[C] Z \leftarrow Z+1
IF Z = 0 GOTO C
[D] X \leftarrow X-1
Y \leftarrow Y+1
IF X \neq 0 GOTO D
Y \leftarrow Y-1
```

Composition

Theorem 1.1: If $h$ is obtained from the (partially) computable functions $f, g_1, \ldots, g_k$ by composition, then $h$ is (partially) computable.

Proof: The following program obviously computes $h$:

```
Z_1 \leftarrow g_1(X_1, \ldots, X_n)
\vdots
Z_k \leftarrow g_k(X_1, \ldots, X_n)
Y \leftarrow f(Z_1, \ldots, Z_k)
```

If $f, g_1, \ldots, g_k$ are not only partially computable but are also total, then so is $h$.

Composition

Example:
We know that $f(x_1, x_2) = x_1 + x_2$ is a computable function.
We also know that $g_1(x) = x^2$ and $g_2(x) = 3x$ are computable functions.

According to Theorem 1.1, the following function $h(x)$ must then also be computable:

$h(x) = f(g_1(x), g_2(x)) = f(x^2, 3x) = x^2 + 3x$
Recursion

Let $k$ be some fixed number and

$h(0) = k$
$h(t + 1) = g(t, h(t))$

where $g$ is some given total function of two variables. Then we say that $h$ is obtained from $g$ by **primitive recursion**, or simply **recursion**.

**Theorem 2.1:** Let $h$ be obtained as shown above, and let $g$ be computable. Then $h$ is also computable.

Proof:

Obviously, the function $f(x) = k$ is computable. The program computing $f(x)$ simply consists of $k$ times the instruction $Y \leftarrow Y + 1$. This gives us the macro $Y \leftarrow k$

Now we can write a program that computes $h(x)$:

```
Y \leftarrow k
[A] IF X=0 GOTO E
    Y \leftarrow g(Z, Y)
    Z \leftarrow Z+1
    X \leftarrow X-1
    GOTO A
```

Recursion

Here is a similar, but more complicated type of recursion:

$h(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n)$
$h(x_1, \ldots, x_n, t + 1) = g(t, h(x_1, \ldots, x_n, t), x_1, \ldots, x_n)$

Here we say that the function $h$ of $n + 1$ variables is obtained by **primitive recursion** (or **recursion**) from the functions $f$ (of $n$ variables) and $g$ (of $n + 2$ variables).

**Theorem 2.2:** Let $h$ be obtained from $f$ and $g$ as shown above, and let $f$ and $g$ be computable. Then $h$ is also computable.

Proof:

The following program computes $h(x_1, \ldots, x_n, x_{n+1})$:

```
Y \leftarrow f(x_1, \ldots, x_n)
[A] IF X_{n+1}=0 GOTO E
    Y \leftarrow g(Z, Y, X_1, \ldots, X_n)
    Z \leftarrow Z+1
    X_{n+1} \leftarrow X_{n+1}+1
    GOTO A
```

Recursion