Solution to Problem 14

**Problem.** Sketch an example of a Turing machine for implementing primitive recursion (i.e., a for-loop), the way we did it in class, on the example of the following simple for-loop

\[
\text{sum} = a; \\
\text{for} (\text{i} = 1; \ i \leq b; \ i++) \\
\{ \text{sum} = \text{sum} + c; \}
\]

No details are required, but any details will give you extra credit.

**Solution.** In mathematical terms, the above for-loop takes the following form:

\[
\text{sum}(a, 0) = a; \\
\text{sum}(a, m + 1) = \text{sum}(a, m) + c.
\]

After we rename the function \( \text{sum} \) into \( h \) and the parameters \( a \) and \( c \) into \( n_1 \) and \( n_2 \), we get the standard form:

\[
h(n_1, n_2, 0) = n_1; \\
h(n_1, n_2, m + 1) = h(n_1, n_2, m) + n_2.
\]

In this standard form, we have \( f(n_1, n_2) = n_1 \), i.e., \( f = \pi_1^2 \), and \( g(n_1, n_2, m, h) = h + n_2 \), i.e., \( g = \text{add}(\pi_4^2, \pi_2^1) \).

Let us follow the general scheme for computing primitive recursion. Suppose that we have Turing machines computing the functions \( f(n_1, n_2) = n_1 \) and \( g(n_1, n_2, m, h) = h + n_2 \). Let us show how to build a Turing machine that compute the desired function \( h = PR(f, g) \). We start with the state

\[
\_ \ n_1 \ - \ n_2 \ - \ x \ - \ \ldots \ \text{start}
\]

and we want to end up in the state

\[
\_ \ h(n_1, n_2, x) \ - \ \ldots \ \text{halt}
\]

This can be done as follows. First, we copy \( x \), add 0, then copy the numbers \( n_1 \) and \( n_2 \), and move the head into the cell right before the second copy of \( n_1 \):

\[
\_ \ n_1 \ - \ n_2 \ - \ x \ - \ x \ - \ 0 \ - \ n_1 \ - \ n_2 \ - \ \ldots
\]
Then, we apply the Turing machine $f$. Since a Turing machine never goes beyond the cell where it starts, it will compute the value 

$$h(n_1, n_2, 0) = f(n_1, n_2) = n_1,$$

so we will have the following state of the tape:

$$- n_1 - n_2 - x - x - 0 - h(n_1, n_2, 0) - ...$$

Now, we copy $n_1$, $n_2$, and 0 before $h$, and get

$$- n_1 - n_2 - x - x - 0 - n_1 - n_2 - 0 - h(n_1, n_2, 0) - ...$$

Then, we apply the Turing machine for computing the function $g$, and get 

$$h(n_1, n_2, 1) = g(n_1, n_2, 0, h(n_1, n_2, 0)).$$

So, the tape has the form:

$$- n_1 - n_2 - x - x - 0 - h(n_1, n_2, 1) - ...$$

After that, we decrease the second copy of $x$ by 1, increase 0 by 1, and get the following:

$$- n_1 - n_2 - x - x - 0 - h(n_1, n_2, 1) - ...$$

and we repeat a similar procedure.

In general, for each $m \leq x$, we get the following state of the tape:

$$- n_1 - n_2 - x - x - m - m - h(n_1, n_2, m) - ...$$

Then, we copy $n_1$, $n_2$, and $m$ before $h$, and get

$$- n_1 - n_2 - x - x - m - m - n_1 - n_2 - m - h(n_1, n_2, m) - ...$$

Now, we apply the Turing machine for computing the function $g$, and get 

$$h(n_1, n_2, m + 1) = h(n_1, n_2, m, h(n_1, n_2, m)).$$

So, the tape has the form:

$$- n_1 - n_2 - x - x - m - m - h(n_1, n_2, m + 1) - ...$$
Then, we check whether \( x - m = 0 \). If \( x - m > 0 \), we decrease \( x - m \) by 1, increase \( m \) by 1, and get the following:

\[
- n_1 - n_2 - x - x - (m + 1) - m + 1 = h(n_1, n_2, m + 1) - \ldots
\]

and we repeat a similar procedure.

Once we get \( x - m = 0 \) i.e., \( m = x \), the state of the tape takes the form

\[
- n_1 - n_2 - x - 0 - x = h(n_1, n_2, x) - \ldots
\]

Here, we have \( k + 4 = 6 \) numbers:

- the numbers \( n_1 \) and \( n_2 \), and
- four numbers \( x, 0, x, \) and \( h(n_1, n_2, x) \).

The desired value \( h(n_1, n_2, x) \) is 6-th out of 6, so we can get it by applying the Turing machine computing the corresponding projection \( \pi_6^6 \):

\[
- h(n_1, n_2, x) - \ldots \text{ halt}
\]

This is exactly what we wanted.

In this construction, we use composition, adding 1, subtracting 1, copying, and projection. We know how to do all this on a Turing machine, so indeed we can thus build a Turing machine for computing the function \( PR(f, g) \).