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

\[
\begin{align*}
v &= a; \\
\text{for} (\text{int } i = 1; i <= b; i++) \\
& \quad \{ v = v * i; \}
\end{align*}
\]

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

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

\[
v(a, 0) = a;
\]

\[
v(a, m + 1) = v(a, m) * (m + 1).
\]

After we rename the function \(v\) into \(h\) and the parameter \(a\) into \(n\), we get the standard form:

\[
h(n, 0) = n;
\]

\[
h(n, m + 1) = h(n, m) * (m + 1).
\]

In this standard form, we have \(f(n_1) = n_1\), i.e., \(f = \pi_1^n\), and \(g(n_1, m, h) = h * (m + 1)\), i.e., \(g = \text{prod}(\pi_3^n, \sigma(\pi_2^n))\).

Let us follow the general scheme for computing primitive recursion. Suppose that we have Turing machines computing the functions \(f(n_1) = n_1\) and \(g(n_1, m, h) = h * (m + 1)\). Let us show how to build a Turing machine that compute the desired function \(h = PR(f, g)\). We start with the state

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

and we want to end up in the state

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

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

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

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

so we will have the following state of the tape:

```
- n1  x  x  x  0  h(n1, 0)  ...
```

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

```
- n1  x  x  x  0  n1  0  h(n1, 0)  ...
```

Then, we apply the Turing machine for computing the function $g$, and get $h(n_1, 1) = g(n_1, 0, h(n_1, 0))$. So, the tape has the form:

```
- n1  x  x  x  0  h(n1, 1)  ...
```

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

```
- n1  x  x  x  0  1  h(n1, 1)  ...
```

and we repeat a similar procedure.

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

```
- n1  x  x  x  m  h(n1, m)  ...
```

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

```
- n1  x  x  x  m  n1  m  h(n1, m)  ...
```

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

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

So, the tape has the form:

```
- n1  n2  x  x  m  h(n1, 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:
and we repeat a similar procedure. Once we get $x - m = 0$, i.e., $m = x$, the state of the tape takes the form

$$- n_1 - x - x - (m + 1) - m + 1 - h(n_1, m + 1) - \ldots$$

Here, we have $k + 4 = 5$ numbers:

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

The desired value $h(n_1, x)$ is 5-th out of 5, so we can get it by applying the Turing machine computing the corresponding projection $\pi_5^5$:

$$- h(n_1, x) - \ldots \text{ halt}$$

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 $PR(f, g)$.