1. $x \dot{-} 1$:

```
w = 0
z = 0
loop(x) {
    w = z
    z++ }
return w
```

$x \dot{-} y$

```
loop(y) {
    w = 0
    z = 0
    loop(x) {
        w = z
        z++ }
    x = w }
return x
```

2. Because no instruction can increase any register by more than 1, and hence at least $g(x) \dot{-} x$ instructions will have to be executed to put the value $g(x)$ in the output register.

3. $f_1(x) = \begin{cases} 1 & \text{if } x = 0 \\ 2x & \text{otw.} \end{cases}$ Alternatively $f = (2x \dot{-} 1) + 1$.

4. We observe that any increasing function $f : \mathbb{N} \to \mathbb{N}$ with $f(0) > 0$ is expansive. For suppose, inductively that $f(x) > x$. Then since $x + 1 > x$ $f(x+1) \geq f(x) + 1 > x + 1$, as required.

Observe also that, for all n, $f_n(0) = 1 > 0$.

We show by induction on n that f_n is increasing (hence expansive). Evidently, f_0 is increasing. Suppose f_n is increasing (hence expansive). Then $f_{n+1}(x+1) = f_n^{(x+1)}(1) = f_n(f_n^{(x)}(1)) > f_n^{(x)}(1) = f_{n+1}(x)$. Hence f_{n+1} is increasing, and therefore expansive.

5. It suffices to show by induction that $f_{n+1}(x) \geq f_n(x)$.

For $n = 0$, it is obvious that $f_1(x) \geq f_0(x)$.

Assuming the result for n, we have

$$f_{n+2}(x) = f_{n+1}(\underbrace{\cdots}_{x \text{ times}}(f_{n+1}(1))\cdots)$$

$$\geq f_n(\underbrace{\cdots}_{x \text{ times}}(f_n(1)))$$

$$= f_{n+1}(x). \quad (\text{since } f_{n+1} \text{ increasing})$$

Assume first that $x > 0$

Now for $n \geq 1$

$$f_n^{(k+1)}(x) = f_n(f_n^{(k)}(x))$$

$$\geq f_1(f_n^{(k)}(x))$$

$$\geq 2 f_n^{(k)}(x) \quad (\text{using } f_n^k(x) \geq x > 0)$$

$$\geq f_n^{(k)}(x) + x \quad \text{since } f_n \text{ is expansive}$$

On the other hand, if $x = 0$, the result follows immediately from the fact that f_n is expansive.

6. $f_1(x) = \begin{cases} 1 & \text{if } x = 0 \\ 2x & \text{ow} \end{cases}$

$= (2x \dot{-} 1) + 1$

This can be computed by

```
loop x {
    x++ }
w1 = 0
z = 0
loop (x) {
    w1 = z
    z++ }
w1++
return w1
```

} double x as in Q1

} compute $x \dot{-} 1$ as in Q 1

and so is in $\mathcal{L}_1$. Suppose $(P_n ;$ return $w_n)$ is a program computing $f_n(x)$ and using registers $w_1, \ldots w_n, y$ putting the return value in w_n. To compute f_{n+1}, execute the program $(P_{n+1} ;$ return $w_{n+1})$ where P_{n+1} is

```
y = 1
w_{n+1} = 1
loop (x) {
    x = y
    P_n
    y = w_n }
w_{n+1} = y
```

If P_n is in $\mathcal{L}_{n-1}$, then P_{n+1} is in $\mathcal{L}_{n+1}$.

7. If running time of P is bounded by $f_n^{(k)}(x)$ then the function g computed by P is bounded by $f_n^{(k)}(x) + x$ hence by $f_n^{(k+1)}(x)$ for $n \geq 1$. But then $g = f_{n+1}$ cannot be computed by a program with at most n nested loops, since f_{n+1} eventually dominates $f_n^{(k+1)}$. For $n = 0$, we simply observe that P can increase x by only a constant amount, hence $f_1 \notin \mathcal{L}_0$.