I use $\cdot$ for unit type (other common variations are: ( ), unit, void).

Also define some types. Note that these are not type variables, it is just for compaction.
Let $t_a = (\text{int} \times (\cdot \rightarrow \text{int}) \times (\cdot \rightarrow \text{int}) \times (\cdot \rightarrow \text{int}) \times (\cdot \rightarrow \text{int})) \rightarrow \text{int}$
Let $t_x = \text{int} \rightarrow (\cdot \rightarrow \text{int})$

We'll also define some typing assumptions, again just to reduce the size of our rules.
Let $\Gamma_4 = \Gamma_3 \bigcup \{b : (\cdot \rightarrow \text{int})\}$
Let $\Gamma_3 = \Gamma_1 \bigcup \{k : \text{int},\ x_1 : (\cdot \rightarrow \text{int}),\ x_2 : (\cdot \rightarrow \text{int}),\ x_3 : (\cdot \rightarrow \text{int}),\ x_4 : (\cdot \rightarrow \text{int})\}$
Let $\Gamma_2 = \Gamma_1 \bigcup \{x : t_x\}$
Let $\Gamma_1 = \{a : t_a\}$

Actual Proof

The proof is broken up into stages, to keep it readable. We begin with a nested (recursive) let-statement, describing the entire program.

$$\text{R}_{\text{letR}}\ \frac{\Gamma_1 \vdash \lambda k x_1 x_2 x_3 x_4.\text{(body of a)} : t_a \qquad \Gamma_1 \vdash \text{let } x = \lambda n.\lambda.n \text{ in } a(10, x(1), x(-1), x(1), x(0)) : \text{int}}{\cdot \vdash \text{let } a = \lambda k x_1 x_2 x_3 x_4.\text{(body of a) in let } x = \lambda n.\lambda.n \text{ in } a(10, x(1), x(-1), x(1), x(0)) : \text{int}}$$

Note that the body of x is expressed as $\lambda.n$, which is a slight abuse of λ-calculus structure to represent a function of no arguments. We'll use the same kind of syntax in the description below, when we encounter the b function.

Now, we prove the two conditions above the line separately.

1 The type of a, as a λ-abstraction

First, we continue the proof until we're into the stage of typing the body of b, and the return value of a, which we will continue as sub-proofs.

$$\text{R}_{\text{abs}}\ \frac{\text{R}_{\text{letR}}\ \dfrac{\Gamma_4 \vdash \lambda.\text{(body of b)} : (\cdot \rightarrow \text{int}) \qquad \Gamma_4 \vdash k > 0\ ?\ b() : x_3() + x_4() : \text{int}}{\Gamma_3 \vdash \text{let } b = \lambda.\text{(body of b) in } k > 0\ ?\ b() : x_3() + x_4() : \text{int}}}{\Gamma_1 \vdash \lambda k x_1 x_2 x_3 x_4.\text{(body of a)} : t_a}$$

1.1 The type of b, as a λ-abstraction

The proof here reduces to proving the types of the two statements of b. For this we need a new rule, in which we type the first statement, and assume the type of the sequence is the type of the second (return) statement.

The overall type of the first statement doesn't matter, as long as it's internally consistent. We could assume it's an arbitrary statement, and let it be a type variable. In many cases statements are assumed to not return any value, in which case their type is unit. Here we know it's an assignment though, and technically an assignment returns the type of the variable assigned, so we'll be specific and assume it's an int.

We'll call our very specific rule "assign;return".

$$\text{R}_{\text{abs}}\ \frac{\text{R}_{\text{assign;return}}\ \dfrac{\Gamma_4 \vdash k -= 1 : \text{int} \qquad \Gamma_4 \vdash a(k, b, x_1, x_2, x_3) : \text{int}}{\Gamma_4 \vdash k -= 1; a(k, b, x_1, x_2, x_3) : \text{int}}}{\Gamma_4 \vdash \lambda.\text{(body of b)} : (\cdot \rightarrow \text{int})}$$

This leaves us with the need to prove the types of our two statements.

1.1.1 The first statement of b

We'll need a new rule for the -= statement. This is trivial, but it should match what our "assign;return" rule assumed it would do.

$$\text{R}_{-=}\frac{\text{R}_{\text{var}}\dfrac{}{\Gamma_4 \vdash k : \text{int}} \qquad \text{R}_{\text{const}}\dfrac{}{\Gamma_4 \vdash 1 : \text{int}}}{\Gamma_4 \vdash k -= 1 : \text{int}}$$

1.1.2 The second statement of b

The actual return statement is just a straightforward use of application. We'll simplify it a little though, to reduce the space required, our var-rule to point out that under Γ_4 we can assume all of b, x_1, x_2, x_3 all have type $(\cdot \to \text{int})$.

$$\text{lemma}_1 = \text{R}_{\text{var}}\frac{}{\Gamma_4 \vdash b : (\cdot \to \text{int})}, \text{ and } \text{R}_{\text{var}}\frac{}{\Gamma_4 \vdash x_{i\in\{1,2,3\}} : (\cdot \to \text{int})}$$

So now we can just use the above claim to finish the sub-proof.

$$\text{R}_{\text{appl}}\frac{\text{R}_{\text{var}}\dfrac{}{\Gamma_4 \vdash a : t_a} \qquad \text{R}_{\text{var}}\dfrac{}{\Gamma_4 \vdash k : \text{int}} \qquad \text{lemma}_1}{\Gamma_4 \vdash a(k, b, x_1, x_2, x_3) : \text{int}}$$

1.2 The return value of a

First, we note that a ternary expression is really just a conditional, so we'll reuse the cond-rule to break it up into 3 pieces.

$$\text{R}_{\text{cond}}\frac{\Gamma_4 \vdash k > 0 : \text{bool} \qquad \Gamma_4 \vdash b() : \text{int} \qquad \Gamma_4 \vdash x_3() + x_4() : \text{int}}{\Gamma_4 \vdash k > 0\ ?\ b() : x_3() + x_4() : \text{int}}$$

We can then show a proof of each of these pieces, first the condition,

$$\text{R}_{>}\frac{\text{R}_{\text{var}}\dfrac{}{\Gamma_4 \vdash k : \text{int}} \qquad \text{R}_{\text{const}}\dfrac{}{\Gamma_4 \vdash 0 : \text{int}}}{\Gamma_4 \vdash k > 0 : \text{bool}}$$

then the then-side,

$$\text{R}_{\text{appl}}\frac{\text{R}_{\text{var}}\dfrac{}{\Gamma_4 \vdash b : (\cdot \to \text{int})}}{\Gamma_4 \vdash b() : \text{int}}$$

and finally the else-side.

$$\text{R}_{+}\frac{\text{R}_{\text{appl}}\dfrac{\text{R}_{\text{var}}\dfrac{}{\Gamma_4 \vdash x_3 : (\cdot \to \text{int})}}{\Gamma_4 \vdash x_3() : \text{int}} \qquad \text{R}_{\text{appl}}\dfrac{\text{R}_{\text{var}}\dfrac{}{\Gamma_4 \vdash x_4 : (\cdot \to \text{int})}}{\Gamma_4 \vdash x_4() : \text{int}}}{\Gamma_4 \vdash x_3() + x_4() : \text{int}}$$

2 Let-abstraction of x containing the actual invocation of a

This is the second part of our starting point. Here we'll use a let-rule to first break it down into the body of x, and the actual invocation. Note that we do not actually need a *recursive* let-rule, as x isn't recursive, although using the recursive rule work fine too.

$$\mathrm{R_{let}}\,\dfrac{\Gamma_1 \vdash \lambda n.\lambda.n : t_x \qquad \Gamma_2 \vdash a(10, x(1), x(-1), x(1), x(0)) : \text{int}}{\Gamma_1 \vdash \text{let } x = \lambda n.\lambda.n \text{ in } a(10, x(1), x(-1), x(1), x(0)) : \text{int}}$$

2.1 The type of x, as a λ-abstraction

This is straightforward. x is a λ-abstraction of a single int-type, returning a λ-abstraction of no arguments to an int.

$$\mathrm{R_{abs}}\,\dfrac{\mathrm{R_{abs}}\,\dfrac{\mathrm{R_{var}}\,\dfrac{}{\Gamma_1, n : \text{int} \vdash n : \text{int}}}{\Gamma_1, n : \text{int} \vdash \lambda \cdot .n : (\cdot \to \text{int})}}{\Gamma_1 \vdash \lambda n.\lambda \cdot .n : t_x}$$

2.2 The final invocation of a

This is also straightforward, an application of constants, and applications to a function of known type.

To reduce the space required, however, we again show a short lemma, in this case proving that under Γ_2, we know $x(c) : (\cdot \to \text{int})$ for any constant c.

$$\text{lemma}_2 = \mathrm{R_{appl}}\,\dfrac{\mathrm{R_{var}}\,\dfrac{}{\Gamma_2 \vdash x : t_x} \qquad \mathrm{R_{const}}\,\dfrac{}{\Gamma_2 \vdash c : \text{int}}}{\Gamma_2 \vdash x(c) : (\cdot \to \text{int})}$$

This lets us prove the final type of the whole thing.

$$\mathrm{R_{appl}}\,\dfrac{\mathrm{R_{var}}\,\dfrac{}{\Gamma_2 \vdash a : t_a} \qquad \mathrm{R_{const}}\,\dfrac{}{\Gamma_2 \vdash 10 : \text{int}} \qquad \text{lemma}_2}{\Gamma_2 \vdash a(10, x(1), x(-1), x(1), x(0)) : \text{int}}$$

Principle Types

$a : (\text{int} \times (\cdot \to \text{int}) \times (\cdot \to \text{int}) \times (\cdot \to \text{int}) \times (\cdot \to \text{int})) \to \text{int}$
$b : (\cdot \to \text{int})$
$x : \alpha \to (\cdot \to \alpha)$