stat	$\longrightarrow$	`if` *bool_expr* `then` *stat* `else` *stat*
	|	*stat* `;` *stat*
	|	*variable* `:=` *expr*
variable	$\longrightarrow$	`x` | `y` | `z` | `w` | `u` | `v`
expr	$\longrightarrow$	*expr* `+` *expr*
	|	*expr* `-` *expr*
	|	*expr* `*` *expr*
	|	*variable*
	|	`0` | `1`
relation	$\longrightarrow$	`<` | `=` | `>`
bool_expr	$\longrightarrow$	*expr relation expr*

Figure 1: Abstract grammar for Semantics Worksheet

Homework Assignment #4

For the sake of concreteness in this assignment assume that we have the abstract grammar shown in Figure 1. Suppose also that the set of locations is given by $\mathcal{L} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{w}, \mathtt{u}, \mathtt{v}\}$, that the set of values that expressions can take is $\mathbb{Z}$. Let S_0 be the memory state $\mathcal{L} \times \{0\} = \{(\mathtt{x}, 0), (\mathtt{y}, 0), (\mathtt{z}, 0), (\mathtt{w}, 0), (\mathtt{u}, 0), (\mathtt{v}, 0)\}$.

1. Suppose that S_1 is the state $\{(\mathtt{x}, 2), (\mathtt{w}, 3), (\mathtt{z}, 1), (\mathtt{y}, 4), (\mathtt{u}, 0), (\mathtt{v}, 0)\}$, and that p is the program

```
x := y+1+1 ; y := 1+1+1+1+1 .
```

What is $\mathcal{C}[\![p]\!](S_0)$? What is $\mathcal{C}[\![p]\!](S_1)$?

$$\mathcal{C}[\![p]\!](S_0) = \{(\mathtt{x}, 2), (\mathtt{w}, 0), (\mathtt{z}, 0), (\mathtt{y}, 3), (\mathtt{u}, 0), (\mathtt{v}, 0)\}, \tag{1}$$

$$\mathcal{C}[\![p]\!](S_1) = \{(\mathtt{x}, 6), (\mathtt{w}, 3), (\mathtt{z}, 1), (\mathtt{y}, 3), (\mathtt{u}, 0), (\mathtt{v}, 0)\}. \tag{2}$$

2. Discuss the claim that for any statement p in our language, we can write

$$\mathcal{C}[\![p]\!](s) = s[\mathtt{x} \mapsto \phi_{\mathtt{x}}(s)][\mathtt{y} \mapsto \phi_{\mathtt{y}}(s)][\mathtt{z} \mapsto \phi_{\mathtt{z}}(s)]$$
$$[\mathtt{w} \mapsto \phi_{\mathtt{w}}(s)][\mathtt{u} \mapsto \phi_{\mathtt{u}}(s)][\mathtt{v} \mapsto \phi_{\mathtt{v}}(s)]$$

for suitable functions $\phi_{\mathtt{x}}$, $\phi_{\mathtt{y}}$, $\phi_{\mathtt{z}}$, , $\phi_{\mathtt{w}}$, $\phi_{\mathtt{u}}$, and $\phi_{\mathtt{v}}$.

All programs halt and never produce errors and always produce the same result for a given state of memory, so $C[\![p]\!](s) = \theta(s)$ for some function θ. For a given state s, $\theta(s)$ has a value for each variable, so we can define functions

$$\phi_{\mathtt{x}}(s) = \theta(s)(\mathtt{x}) \qquad \phi_{\mathtt{y}}(s) = \theta(s)(\mathtt{y}) \qquad \phi_{\mathtt{z}}(s) = \theta(s)(\mathtt{z})$$
$$\phi_{\mathtt{w}}(s) = \theta(s)(\mathtt{w}) \qquad \phi_{\mathtt{u}}(s) = \theta(s)(\mathtt{u}) \qquad \phi_{\mathtt{v}}(s) = \theta(s)(\mathtt{v})$$

It's now just a matter of boring computation to show that these are the right functions. For instance we have

$$\begin{aligned}
\mathcal{C}[\![p]\!](s)(\mathtt{w}) &= s[\mathtt{x} \mapsto \phi_{\mathtt{x}}(s)][\mathtt{y} \mapsto \phi_{\mathtt{y}}(s)][\mathtt{z} \mapsto \phi_{\mathtt{z}}(s)] \\
&\quad [\mathtt{w} \mapsto \phi_{\mathtt{w}}(s)][\mathtt{u} \mapsto \phi_{\mathtt{u}}(s)][\mathtt{v} \mapsto \phi_{\mathtt{v}}(s)](\mathtt{w}) \\
&= s[\mathtt{x} \mapsto \phi_{\mathtt{x}}(s)][\mathtt{y} \mapsto \phi_{\mathtt{y}}(s)][\mathtt{z} \mapsto \phi_{\mathtt{z}}(s)] \\
&\quad [\mathtt{w} \mapsto \phi_{\mathtt{w}}(s)][\mathtt{u} \mapsto \phi_{\mathtt{u}}(s)](\mathtt{w}) && \text{because } \mathtt{w} \neq \mathtt{v} \\
&= s[\mathtt{x} \mapsto \phi_{\mathtt{x}}(s)][\mathtt{y} \mapsto \phi_{\mathtt{y}}(s)][\mathtt{z} \mapsto \phi_{\mathtt{z}}(s)] \\
&\quad [\mathtt{w} \mapsto \phi_{\mathtt{w}}(s)](\mathtt{w}) && \text{because } \mathtt{w} \neq \mathtt{u} \\
&= \phi_{\mathtt{w}}(s) && \text{because } \mathtt{w} = \mathtt{w} \\
&= \theta(s)(\mathtt{w}) && \text{by definition}
\end{aligned}$$

3. Suppose that we extend the abstract syntax of statements so that we can write empty statements, for instance, `;;;x:=1+1;;`. What should the meaning of the empty statement be? The meaning of a statement should be a function from states to states that does nothing, that is the identity function:

$$C[\![;;]\!](S) = S. \tag{3}$$

4. Completely write out the formal denotational semantics for the programming language shown in Figure 1 using functions $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{E}$, where the domains and co-domains are specified as follows:

function	domain	co-domain
$\mathcal{C}$	*stat*	$\mathcal{S} \mapsto \mathcal{S}$
$\mathcal{E}$	*expr*	$\mathcal{S} \mapsto \mathbb{Z}$
$\mathcal{B}$	*bool_expr*	$\mathcal{S} \mapsto \{\mathsf{T}, \mathsf{F}\}$

In the following equations s_1 and s_2 are meta-syntactic variables that range over statements (*stat*); e_1 and e_2 are meta-syntactic variables that range over expresssions (*expr*); and v_1 is a meta-syntactic variable that ranges over $\{\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{u}, \mathtt{v}\}$ (*variable*). We let S stand for a memory state, that is a function in $\mathbb{Z}^{\{\mathtt{x},\mathtt{y},\mathtt{z},\mathtt{u},\mathtt{v}\}}$.

$$\mathcal{C}[\![\,\mathtt{if}\ e_1\ \mathtt{then}\ s_1\ \mathtt{else}\ s_2]\!](S) = \begin{cases} \mathcal{C}[\![s_1]\!](S) & \text{if } \mathcal{B}[\![e_1]\!](S) = \mathsf{T} \\ \mathcal{C}[\![s_2]\!](S) & \text{if } \mathcal{B}[\![e_1]\!](S) = \mathsf{F} \end{cases} \tag{4}$$

$$\mathcal{C}[\![s_1\,;s_2]\!] = \mathcal{C}[\![s_2]\!] \circ \mathcal{C}[\![s_1]\!] \tag{5}$$

$$\mathcal{C}[\![v_1 \ \mathtt{:=}\ e_1]\!](S) = S[v_1 \mapsto \mathcal{E}[\![e_1]\!](S)] \tag{6}$$

$$\mathcal{E}[\![e_1 \,\mathtt{+}\, e_2]\!](S) = \mathcal{E}[\![e_1]\!](S) + \mathcal{E}[\![e_2]\!](S) \tag{7}$$

$$\mathcal{E}[\![e_1 \,\mathtt{*}\, e_2]\!](S) = \mathcal{E}[\![e_1]\!](S) \times \mathcal{E}[\![e_2]\!](S) \tag{8}$$

$$\mathcal{E}[\![e_1 \,\mathtt{-}\, e_2]\!](S) = \mathcal{E}[\![e_1]\!](S) - \mathcal{E}[\![e_2]\!](S) \tag{9}$$

$$\mathcal{E}[\![v]\!](S) = S(v) \tag{10}$$

$$\mathcal{E}[\![\mathsf{1}]\!](S) = 1 \tag{11}$$

$$\mathcal{E}[\![\mathsf{0}]\!](S) = 0 \tag{12}$$

$$\mathcal{B}[\![e_1 \,\mathtt{<}\, e_2]\!](S) = \begin{cases} \mathsf{T} & \text{if } \mathcal{E}[\![e_1]\!](S) < \mathcal{E}[\![e_2]\!](S), \\ \mathsf{F} & \text{otherwise} \end{cases} \tag{13}$$

$$\mathcal{B}[\![e_1 \,\mathtt{=}\, e_2]\!](S) = \begin{cases} \mathsf{T} & \text{if } \mathcal{E}[\![e_1]\!](S) = \mathcal{E}[\![e_2]\!](S), \\ \mathsf{F} & \text{otherwise} \end{cases} \tag{14}$$

$$\mathcal{B}[\![e_1 \,\mathtt{>}\, e_2]\!](S) = \begin{cases} \mathsf{T} & \text{if } \mathcal{E}[\![e_1]\!](S) > \mathcal{E}[\![e_2]\!](S), \\ \mathsf{F} & \text{otherwise} \end{cases} \tag{15}$$