In this assignment, you will improve your understanding of SMoL using the output from the Stacker. In every case, you will be shown a trace configuration and asked to perform some interpretive task. You are welcome to run the Stacker yourself if it will help you construct answers. Also remember that you can "turn off" the Stacker's tracing in two ways:

1. Add the `#:no-trace` option (see more below).
2. Remove the `stacker/` prefix to run under the corresponding `smol` language.

Pro-tip for using `#:no-trace`: have your top two lines look like this:

```
#lang stacker/smol/fun
; #:no-trace
```

That way it's commented out by default. Any time you want to turn off the tracing, uncomment that line (just remove the `;`), and re-comment when you want tracing back on.

In this assignment, we will have two kinds of activities:

1. Given a configuration, *construct a program* that could have produced that trace. Ideally we want an *exact* match (except, of course, for the random addresses), but get as close as you can.
2. Given a configuration, *determine the value of the program* from the trace.

For all programs, assume that the program contained a function `pause`:

```
(deffun (pause) 0)
```

(The programs are written to only use additive arithmetic, so that the result of `pause` will not affect the result.) Imagine that the call to `pause` is about to finish at the point where the configuration was captured.

Configuration → Program

In this portion, we will show you two Stacker trace configurations. For each one, you should write a program that, when run, will result in this configuration. The first trace is from

```
#lang stacker/smol/fun
```

and the second from

```
#lang stacker/smol/state
```

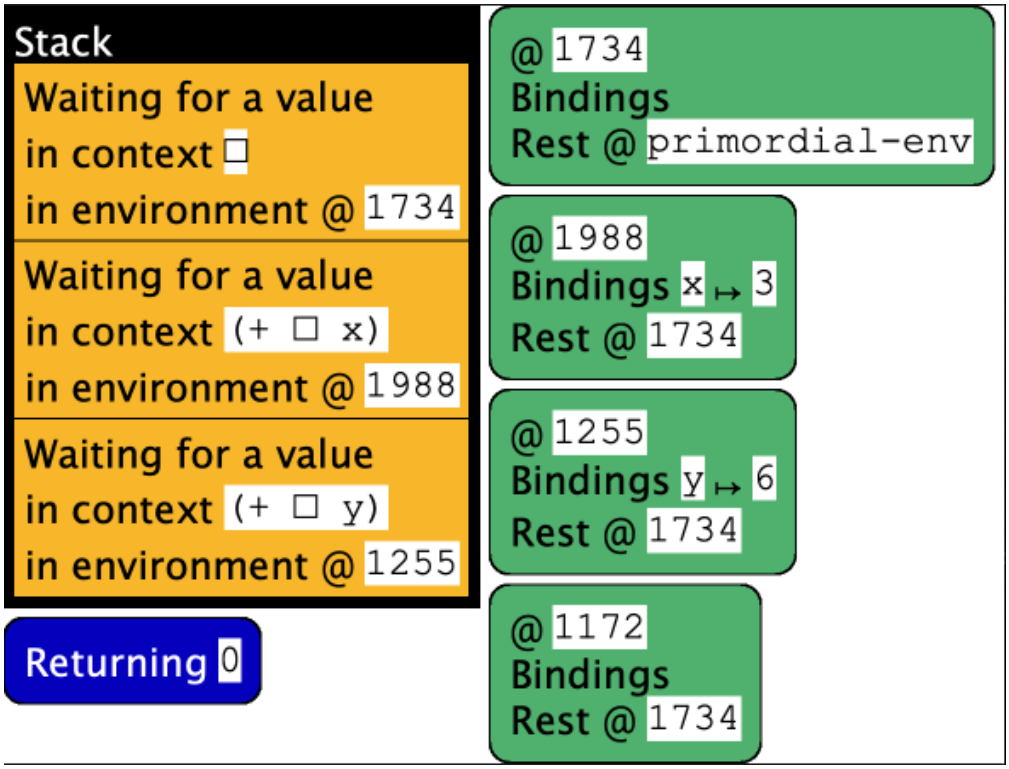
but you can write both in the latter language if you want: the output would be the same.

In each problem, the "Returning `0`" and the bottommost environment are from `pause`. You should include `pause` in the programs that you construct.

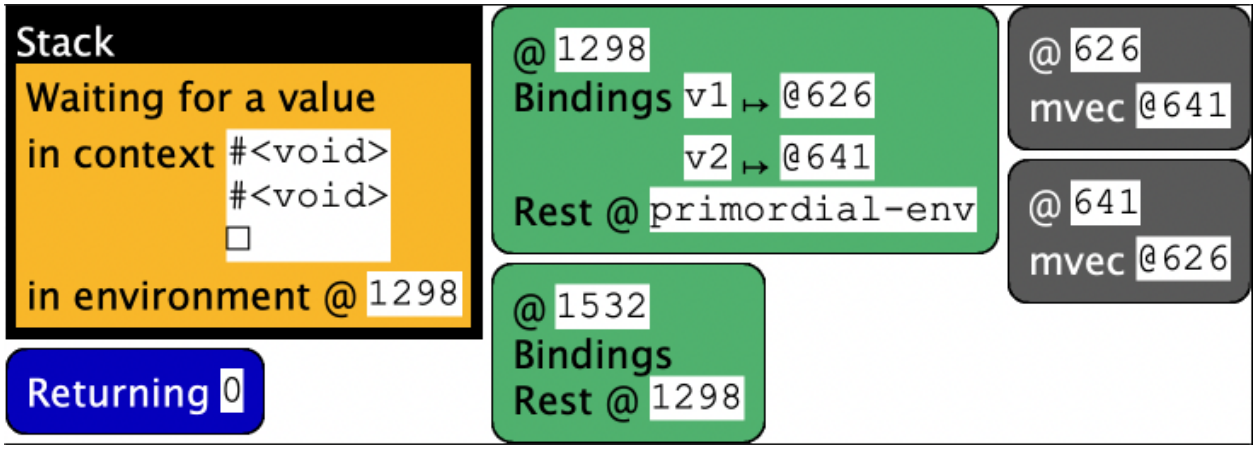
As you might have guessed, infinitely many programs could have produced each configuration. Therefore, there is not only one "correct" answer. We do ask that you try to produce a reasonably minimal solution and avoid flights of fancy. If you really want to get clever, then also include a simple solution!

(There are two problems, one each on the two successive pages.)

1.


Stack
Waiting for a value
in context □
in environment @ 1734
Waiting for a value
in context (+ □ x)
in environment @ 1988
Waiting for a value
in context (+ □ y)
in environment @ 1255
Returning 0
@ 1734
Bindings
Rest @ primordial-env
@ 1988
Bindings x ↦ 3
Rest @ 1734
@ 1255
Bindings y ↦ 6
Rest @ 1734
@ 1172
Bindings
Rest @ 1734

2.


Stack
Waiting for a value
in context #<void>
#<void>
□
in environment @ 1298
Returning 0
@ 1298
Bindings v1 ↦ @626
v2 ↦ @641
Rest @ primordial-env
@ 1532
Bindings
Rest @ 1298
@ 626
mvec @641
@ 641
mvec @626

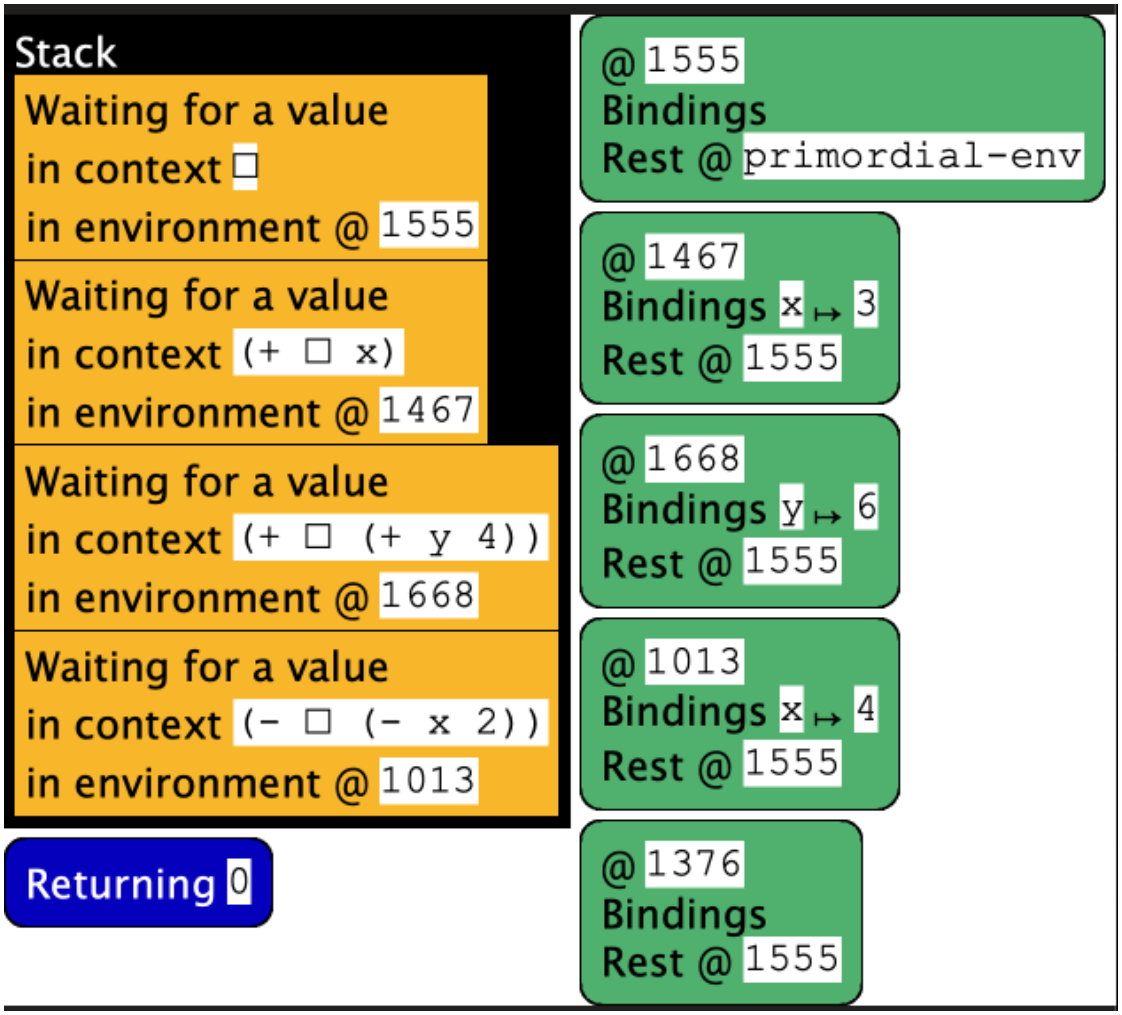
Configuration → Value

In this part, we will show you two configurations and ask you to determine what value the program will produce. Recall that `pause` is designed to not impact the answer.

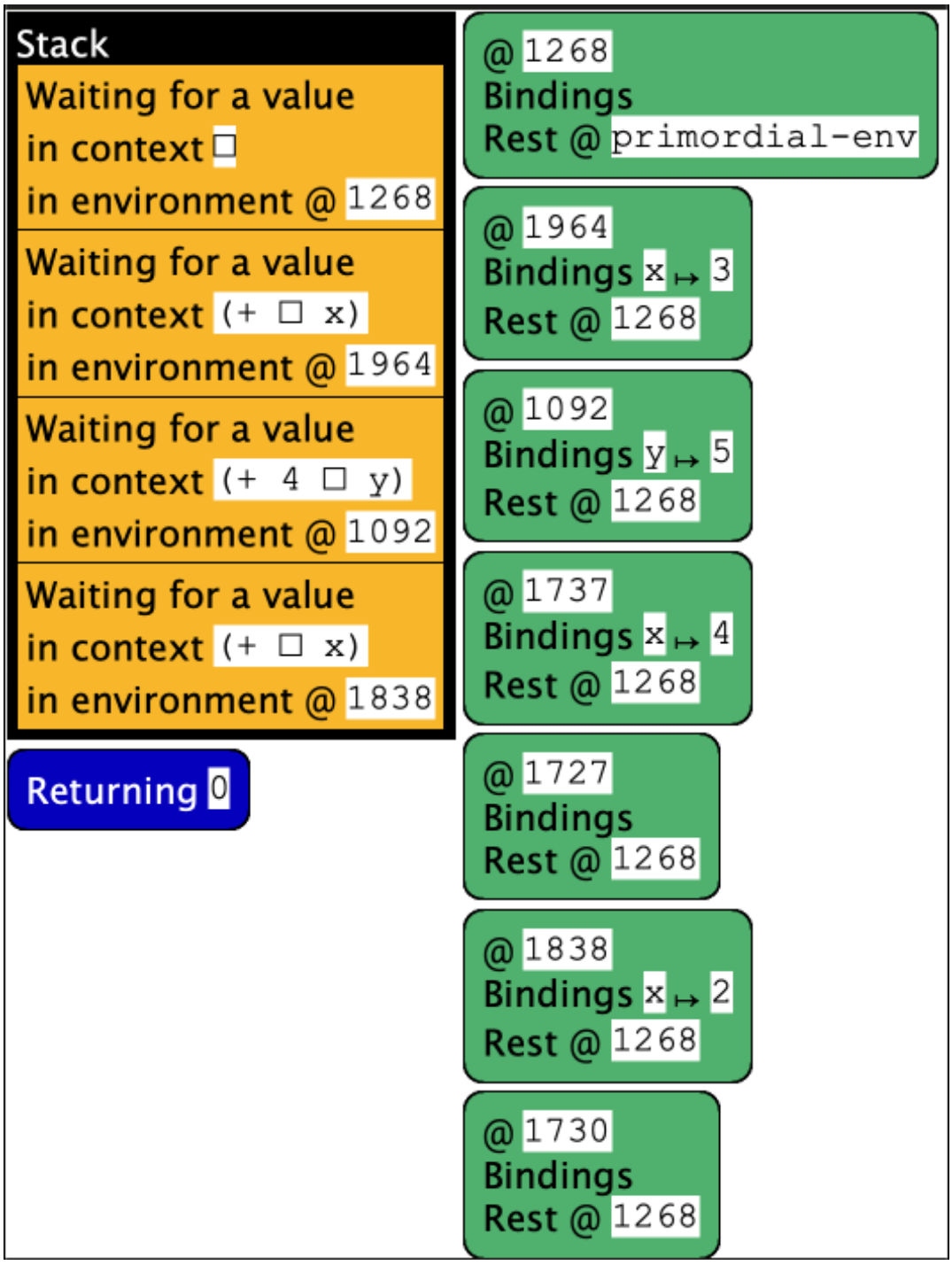
In your response, as your school math teachers used to say, “show your work”. Don’t just give us the answer (a number) but give us a sense of how you arrived at it. You don’t have to be very verbose, just enough to confirm that you understand the mapping from configurations to program results.

(There are two problems, one each on the two successive pages.)

3.


Stack
Waiting for a value
in context □
in environment @ 1555
Waiting for a value
in context (+ □ x)
in environment @ 1467
Waiting for a value
in context (+ □ (+ y 4))
in environment @ 1668
Waiting for a value
in context (- □ (- x 2))
in environment @ 1013
Returning 0
@ 1555
Bindings
Rest @ primordial-env
@ 1467
Bindings x ↦ 3
Rest @ 1555
@ 1668
Bindings y ↦ 6
Rest @ 1555
@ 1013
Bindings x ↦ 4
Rest @ 1555
@ 1376
Bindings
Rest @ 1555

4.


Stack
Waiting for a value
in context □
in environment @ 1268
Waiting for a value
in context (+ □ x)
in environment @ 1964
Waiting for a value
in context (+ 4 □ y)
in environment @ 1092
Waiting for a value
in context (+ □ x)
in environment @ 1838
Returning 0
@ 1268
Bindings
Rest @ primordial-env
@ 1964
Bindings x ↦ 3
Rest @ 1268
@ 1092
Bindings y ↦ 5
Rest @ 1268
@ 1737
Bindings x ↦ 4
Rest @ 1268
@ 1727
Bindings
Rest @ 1268
@ 1838
Bindings x ↦ 2
Rest @ 1268
@ 1730
Bindings
Rest @ 1268