`reset` delimits the part to be "`shifted`"

"`shift`"

```
(* 2 (reset (+ 1 (shift k (k 4)))))
```

→

```
(* 2 (reset (reset (+ 1 4))))
```

New!

→

```
(* 2 (reset (reset 5)))
```

→

```
(* 2 (reset 5))
```

`reset` ting a value produces just the value

→

```
(* 2 5)
```

→

```
10
```

Trivial shift/reset example

Example from hw

`reset` delimits the part to be "`shifted`"

```
(f (list 1 2 3) (list 9 12 154))
```

→

```
(reset (list (+ (one-of l1) (one-of l2))))
```

with l1 := `(list 1 2 3)`, l2 := `(list 9 12 154)`

"`shift`"

→

```
(reset (… (shift k (apply append (map (lambda (x) (k x)) l1))) …))
```

→

```
(reset (apply append (map (lambda (x) (reset (list (+ x (one-of l2))))) l1)))
```

→

```
(reset (apply append (cons (reset (list (+ 1 (one-of l2))))
                           (map (lambda (x) (reset (list (+ x (one-of l2)))))
                                (list 2 3)))))
```

(example continued)

```
(reset (apply append (cons (reset (list (+ 1 (one-of l2)))) (map …))))
```

→

```
(reset (apply append (cons (reset
(… (shift k (apply append (map (lambda (x) (k x)) l2)) …)
) (map …))))
```

→

```
(reset (apply append (cons (reset
(apply append (map (lambda (x) (reset (list (+ 1 x)))) l2))
) (map …))))
```

→*

```
(reset (apply append (cons
(list 10 13 155)
(map …))))
```

Further reading (not relevant for exam):

http://okmij.org/ftp/continuations/against-callcc.html