Parameter Passing

Function Parameters

```
(define (add5 val) (+ val 5))

(add5 100)
(add5 (* 2 25))
```

actual parameter:
The expression for the parameter in the function invocation.
E.g. `100` and `(* 2 25)`

formal parameter:
The name of the parameter inside the function.
E.g. `val`

Parameter Passing

Call by ...

value: Value copied to formal parameter on function invocation.
result: Formal parameter is uninitialized. Its final value is copied to actual parameter on function return.
value-result: A combination of *value* and *result*.
reference: Formal parameter becomes *alias* for actual parameter.
name: Actual parameter is re-evaluated every time formal parameter is accessed in function.

C, Scheme, and Java are *call by value* (with pointer semantics).

C++ is has *call by value* and *call by reference*.

Call by copy

Call by *value*, *result*, and *value-result* are special cases of *call by copy*.

In *call by copy*, function gets its own copy of variable.

value: Copy value in on function invocation.

result: Copy value out on function return.

value-result: Copy value in on invocation and out on return.

Actual Params & LValues

Call by *result*, *value-result*, and *reference* all require that actual parameter is an *LValue*.

Recall: *LValue* – expression on left hand side of assignment.

Given:

```
int i,j;
int[] array = new int[10];
```

Which of the following are valid *LValues*?

1. `i`
2. `array[i]`
3. `5`
4. `array[array[5]]`
5. `(array[i] + j)`

Example 1

Compare results for using call by *value*, *value-result*, or *reference*:

```
begin
   integer n;

   procedure p(k: integer);
      begin
         n := n+1;
         k := k+4;
         print(n);
      end;

   n := 0;
   p(n);
   print(n);
end;
```


Example 2

Compare results for using call by *reference* or *name*:

```
begin
   array a[1..10] of integer;
   integer n;

   procedure p(b: integer);
      begin
         print(b);
         n := n+1;
         print(b);
         b := b+5;
      end;

   a[1] := 10;
   a[2] := 20;
   a[3] := 30;
   a[4] := 40;
   n := 1;
   p(a[n+2]);
   new_line;
   print(a);
end;
```