CIS 425—Assignment #7
due in class, Tuesday November 21

---

Problems

1. Parameter Passing

Consider the following procedure, written in an Algol/Pascal-like notation:

```algol
proc power(x, y, z:int)
begin
    z := 1
    while y > 0 do
        z := z*x
        y := y-1
    end
end
```

The code that makes up the body of `power` is intended to calculate $x^y$ and place the result in $z$. However, depending on the actual parameters, `power` may not behave correctly for certain combinations of parameter-passing methods. For simplicity, we only consider call-by-value and call-by-reference.

(a) Assume $a$ and $c$ are assignable integer variables with distinct L-values. Which parameter-passing methods make $c = a^a$ after a call $power(a, a, c)$. You may assume that the R-values of $a$ and $c$ are non-negative integers.

(b) Suppose that $a$ and $c$ are formal parameters to some procedure $P$, and that the expression $power(a, a, c)$ above is evaluated inside the body of $P$. If $a$ and $c$ are passed to $P$ by reference, and become aliases, then what parameter passing method(s) will make $c = a^a$ after a call $power(a, a, c)$? If, after the call, $c = a^a$, does that mean that $power$ actually performed the correct calculation?

Note: In part (a), $a$ and $c$ refer to distinct memory locations, while in part (b), $a$ and $c$ refer to exactly the same memory location. Your job is to find the appropriate parameter passing conventions for `power()` given each scenario.

2. Pass by Value Result

In pass-by-value-result, also called call-by-value-result and copy-in/copy-out, parameters are passed by value, with an added twist. More specifically, suppose a function $f$ with a pass-by-value-result parameter $u$ is called with actual parameter $v$. The activation record for $f$ will contain a location for formal parameter $u$ that is initialized to the R-value of $v$. Within the body of $f$, the identifier $u$ is treated as an assignable variable. On return from the call to $f$, the actual parameter $v$ is assigned the R-value of $u$.

The following pseudo-Algol code illustrates the main properties of pass-by-value-result.

```algol
var x : integer;
x := 0;
procedure p(value-result y : integer)
begin
    y := 42;
x := 5;
end;
p(x);
```
(a) With pass-by-value-result, what is the the final value of x after the call to p(x)? What would the final value of x be if it were passed by reference instead? By value?
   In all three cases, justify your answer.

(b) Translate the program fragment above into ML (or pseudo-ML if you prefer) in a way that makes the operations on locations, and the differences between L-values and R-values, explicit. Your solution should have the form

   val x = ref 0;
   fun p(y':int ref) =
     ...
     p(x);

   Note that in ML, like C and unlike Algol or Pascal, a function may be called as a procedure (in fact, a “procedure” is just a function whose return type is void/unit.

Note: If your code works by returning a value, it’s wrong. It should mimic the behavior of the pseudo-Algol. Your job here is to implement a version of p() with the same black-box behavior as that of the “Algol” program, under different passing conventions. I.e., same output type, same side-effects. Note that the pseudo-Algol works by side effect only, and has "void" or "unit" return type.

3. ........................................ Pass-by-Value/Pass-By-Reference

   The following Pascal program includes the function declaration, P. In this declaration, X is passed by reference and Y is passed by value. What is the value which assigned to Z in this program?:

   function main():integer;
   var K,L,Z: integer;
   function P(var X:integer; Y:integer):integer
   begin
     K:=3;
     L:=5;
     P:=X+Y { In C, this would be "return X+Y;" }
   end P
   begin
     K:=1;
     L:=1;
     Z:=P(K,L)
   end

4. ............................................. Call-by-Name

   The following Pascal function ink was designed to return the value of its argument plus one.

   function ink(x: integer):integer;
   var y: integer;
   begin
     y := 1;
     ink := x + y;
   end

   Now suppose we use a call-by-name convention for the parameter x, but without mandatory renaming. Give an example to illustrate how it might fail.