Problem 1

Explain the inferred type of each of the following declarations? Use the notation that we used in Recitation 9 (Type systems and type inference).

(a) `fun a(x,y) = x + 2 * y;`
(b) `fun b(f,x) = f(f(x));`
(c) `fun c(x,y,b) = if b(y) then x else y;`
(d) `fun d(g,h) = g(3) + h(4);`
(e) `fun e(g,h) = h(g(false));`

Be sure to write a short explanation to show that you understand why the function has the type you give.

Problem 2

For the following Algol/Pascal-like program, write the number printed by running the program under each of the listed parameter passing mechanisms. Explain your answers.

```plaintext
begin
    integer i;
    procedure pass(x,y);
        integer x, y; // types of the formal parameters
        begin
            x := x + 1;
            y := x + 1;
            x := y;
            i := i + 1
        end
    i := 1;
```
pass (i, i);
print i
end

(a) pass-by-value
(b) pass-by-reference

Problem 3
Consider the following procedure, written in an Algol/Pascal-like notation:

\[
\text{proc power(x, y, z : int)}
\text{begin}
\quad z := 1;
\quad \text{while } y > 0 \text{ do}
\quad \quad z := z \times x;
\quad \quad y := y-1;
\text{end}
\text{end}
\]

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 that \(a\) 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 nonnegative integers. Explain your answer.

(b) Suppose that \(a\) and \(c\) are formal parameters to some procedure \(P\) and that the preceding expression power(a, a, c) 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, \(c = a^a\), then does that mean that power actually performed the correct calculation? Explain your answers.

Problem 4
Consider the following program fragment, written pseudo-C:

```
1 int x = 2; {
2   int f (int y) { return x + y; } {
3       int x = 7; {
4         x +
5         f(x);
6     }
7   }
8 }
```

This program would be legal in a version of C with nested functions.

(a) Under static scoping, what is the value of \(x + f(x)\) in this code? During the execution of this code, the value of \(x\) is needed three different times (on lines 2, 4, and 5). For each line where \(x\) is used, state what numeric value is used when the value of \(x\) is requested and explain why these are the appropriate values under static scoping.

(b) Under dynamic scoping, what is the value of \(x + f(x)\) in this code? For each line in which \(x\) is used, state which value is used for \(x\) and explain why these are the appropriate values under dynamic scoping.