Problem 1 (5pts). [From Scott]. Consider the following C declaration, compiled on a 32-bit Pentium machine:

```
struct {
    int n;
    char c;
} A[10][10];
```

If the address of `A[0][0]` is 1000 (decimal), what is the address of `A[3][7]`?

Problem 2 (10pts). Write a C program that uses the union type to determine the endian-ness of the machine. Explain your answer.

Problem 3 (5pts). Assume that your programming language allows array declarations of the form: `[1..10] [1..5,1..5] int a;` (a is an array of 10 2-dimensional arrays of integers). Assume that arrays are stored in column major order. Show the formula for the address of element `a[j][k,l]`. State any assumptions that you make to obtain this formula.

Problem 4 (12pts). Consider the Pascal-like code for function `compute`.

```
double compute(first : integer /*by value*/, last : integer /*by value*/,
                incr : integer /*by value*/, i : integer /*by name*/,
                term : double /*by name*/)
```

result : double := 0.0
i := first
while i <= last do
    result := result + term
    i := i + incr
endwhile
return result

a. (2pts) What is returned by call `compute(1, 10, 1, i, A[i])`?

b. (2pts) What is returned by call `compute(1, 5, 2, j, 1/A[j])`?

c. (2pts) `compute` is a classic example of Jensen’s device, a technique that exploits call by name and side effects. In one sentence, explain what is the benefit of Jensen’s device.

d. (6pts) Write `max`, which uses Jensen’s device to compute the maximum value in a set of values based off of an array `A`.

Problem 5 (8pts). [Modified from Scott]. Is it possible to write a `swap` routine in Java, or in any other language that uses the reference model for variables and thus only call-by-sharing? What exactly should `swap` do in such a language? (Hint: think about the distinction between the object to which a variable refers and the value (contents) of that object.)