CSE505 – Fall 2012
Assignment 1 – Procedural Languages
Assigned Mon, Sept 10
Due, Weds, Sept 19

1 [20%]. Fortran 77 and earlier versions exhibited 'history-sensitive' behavior because a local variable retained its value from the previous invocation. Show how this behavior can be exploited to define a subroutine, RAND, without any formal parameters and of result type INTEGER , that generates a sequence of pseudo-random integers. Show Fortran code for RAND and briefly explain its behavior.

2 [40%]. Consider the following program in a statically-scoped language:

```
program main
  int p, q;
  void A(value int q, void C(var int)) {
    void B(var int p) {
        p := q + 50;  print(p);   C(q);
    }
    if (p == q) then C(q) else A(q+25, B);
  }
  void D(var int p) {
       p := q + 100;    print(p);
  }
  p := 100;    q := 100;      A(50, D);
end main.
```

Draw a contour diagram showing the nesting structure of all contours when a print statement is executed for the second time. Show the entire sequence of values printed by the above program.

3 [20%] Assuming that the *sigma* procedure discussed in Lecture 3 is translated in terms of function parameters (in the place of name parameters), show how one should translate the following expression. In particular, show the *thunks* that would need to be created.

$$\sum_{i=1}^{n} \; \sum_{j=1}^{n} (\, b[i, j] * \; \sum_{k=1}^{n} c[i, j, k] \,)$$

4 [20%]. What is output produced by the following C program when compiled with the gcc compiler? (Ignore any compiler warnings.) Based upon this output what do you infer about the C language's rule for assignment (copying or sharing) and rule for object allocation (static, quasi-dynamic, or fully dynamic)?

```
#include <stdio.h>
int* foo () {
    int a[3] = {1,10,100};
    return a;
}
int* goo () {
    int b[3] = {2,20,200};
    return b;
}
```

```
int main() {
    int *a, *b
    a = foo();
    b = goo();
    a = printf("%d\n", *a);
    return 0;
}
```

End of Assignment 1