CSE305, Spring 2007 **Problem Set 5** **Due Thu. Feb 22**

The **First Prelim Exam** will be on **Thursday, March 1**, *in class period*. The exam will cover Sebesta chapters 1–7 excepting skimmed parts, and its questions will be drawn from the domain of Assignments 1–5.

This assignment refers to the "Stack-Language" handout given out on Wed. Oct. 5. This is analogous to the Java Virtual Machine, but ignores differences among numeric types and omits many JVM operations. Read this and the files `EvalOrder.cc` and `EvalOrder.java` (currently in ∼`regan/cse305/LANGUAGES/C++/LECREC07`) for Friday. For next week, read all of chapters 6 and 7.

(1) Trace the execution of the following stack-model program. Show the contents of the stack after each operation, and indicate the final contents of the variables when the computation ends. (9 pts.)

```
x 3 store pop y 7 z x fetch 6 - store * z fetch + store pop
```

(2) Translate the following program into the stack-language model, and trace its execution to verify that you get the same results as on Problem Set 4. (15 + 9 = 24 pts.)

```
#include<stdio.h>
int main(void) {
    int w,z;
    int *p; int*q;

    w = z = 15;
    p = &z;
    q = p;
    *p = (*q) + w;
    w = 3*z - (w + *q);
    printf("w = %d and z = %d and *p = %d and *q = %d\n", w,z,*p,*q);
}
```

(3) The following program is written in a mythical language "A" that preceded Ken Thompson's "B," which preceded C.

```
PROC MAIN;
    k,x: int;
    FUNC A(y: int): int;
        FUNC B(x: int): int;
        Begin
            RETURN(x*(y+1));
        End B;
        FUNC C(y:int): int;
            k: int;
        Begin
            k := 2*y;
            RETURN(B(k));
        End C;
    /* Main body of A begins here */
    Begin
```

```
        if y = 1 then RETURN C(3*y);
                  else RETURN A(y-1);
        endif;
    End A;
    PROC D(x: int);
    Begin
        Print("Answer is: %d\n",x+k);
    End D;
/* Body of MAIN begins here. */
Begin
    k := 3;
    x := A(k);
    D(x);
End MAIN;
```

(a) For each of the four sub-programs, draw up a table showing which of the three variable names `k`, `x`, and `y` are visible within that sub-program, and if so, which block it belongs to (i.e., was declared in). For example, the table for MAIN is: "`k = MAIN.k, x = MAIN.x, y` is invisible." (6 pts.)

(b) Trace out the allocation of stack frames during the execution of this program, assuming static scoping. Show the static and dynamic links from each frame, and work out the final values printed by this program. (You don't need to show as much detail as the text does in Chapter 10—it's enough to show the links and the storage objects in each frame together with their values, in the style of lecture notes. 9 pts.)

(c) Modify your answer to (b) (reviewing part (a) if necessary) in case the language uses dynamic scoping. (6 pts.) item[(d)] In your opinion, (when) is it a good idea to make a function/procedure/method modify visible "global variables" or "class variables" directly, rather than make them explicit parameters to the method? Mention Java's rules that parameters and local variables can "shadow" class fields, but cannot shadow other locals or parameters. (9 pts., for 30 total)

(4) Write in ML a function $f(x)$ that returns $x(x-1)/2$, but note that you will have to use `div` not `/` for division—see why? Then write a function `sumf` that calls your f so that on input $n \geq 1$, it outputs the sum of $f(k)$ from $k = 1$ to n. Note that ML has no for-loop, so you will use recursion for `sumf`. Please write your answers *in hardcopy only*—we will do online submissions of ML programs later. The BNF snippet on Assignment 3 has all the syntax you need. (6+6 = 12 pts., for 75 total on the set)