The **Second Prelim Exam** will be **in-class on Friday, Nov. 16**, yes a week from Friday. It will cover Chapters 1–10 and 15, not including Lisp and skipped sections, and assignments 1–10. You will get the answer key to this assignment **after** the end of class on Wed. 11/14, and the exam will reflect that we will likely be unable to grade your work before then. (Translation: Aspects of this set that will be involved on the exam have been seen before.)

**Reading:** Next week will introduce Prolog, so please read Chapter 16. Also (re-)read Chapter 10 but ignore “displays” and the section on implementing dynamic scoping. Then look ahead to Chapter 11. This opens the way to issues in object-oriented programming language design, which will occupy the remainder of the course. This set is for **both hardcopy and on-line submission**, the latter part in a file CSE305ps10NNN.acl (Allegro Common Lisp, not Scheme).

(1) Code `sumOfSquares` and `ascenders` in Allegro Common Lisp, in a file `ps10NNN.acl`, where as usual NNN are your initials. Please submit both hardcopy and online. (6 + 9 = 15 pts.)

(2) The following program is written in MAdCaL, a language that gives you the power of C and Ada inside ML. One of the innovations of MAdCaL is that the parameter-passing behavior of all functions in a compilation unit is specified by a command-line argument to the compiler! This makes it possible for programmers to write multiple, differently-behaving programs with the same code, thus taking code re-use to new depths!

```ml
#import Madcal.io and then use mouth
//mouth = Madcal output history

fun swap(a,b) return void is
  val temp := a
  begin
    a := b and then b := temp
  end swap;

fun main() is
  val value = 2
  val array = [1,3,5,7,9]
  begin
    swap(value, array[value]) //"array" is indexed from 0.
    mouth << "Value is " + value + "and final array is " + array
  end main;

This code is identical to the code in Sebesta 7, ch. 9, problem 5 on page 423, with the first two calls to swap deleted. (So you can ignore the “MAdCaL” satire and regard this as an edited problem out of the text, with call-by-name added.) Show the final output (or “bomb” if the program throws an unhandled exception) if this program is built using (a) call-by-value, (b) call-by-reference, (c) call-by-name, and (d) call-by-value-return. (3 + 3 + 6 + 6 = 18 pts.)

(3) Text, chapter 9, exercise 7 on page 424. (Typo: `fun` has a missing closing curly brace. 9 pts.)

(4) Translate the following two C/C++ functions into ML. Use recursion to simulate a `for`-loop or `while`-loop, and an ML `list` to simulate an array. Have ML return the whole list (as implemented, it is a pointer anyway). You must use tail recursion and accumulator-passing style. In (b) you must make `f` a parameter, i.e. write a higher-order function. (18 pts. total)

```ml
int* prefixSums(int* a, int n) { /* n is the length of the array a */
  for (int i = 1; i < n; i++) {
    a[i] += a[i-1]; /* LOOP INV: a[i] holds sum of original */
  }
  /* array entries up to i. */
}
```
return a; /* return pointer to modified array */
}

int f (int x) {... unspecified ...} //make f a *parameter* in the ML code.

int numIterations(int arg, int stop) {
    int count = 0;
    while (arg != stop) {
        arg = f(arg);
        count++;
    }
    return count;
}

(5) 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 := 3*y;
        RETURN(B(k));
    End C;
    /* Main body of A begins here */
    Begin
        if y = 1 then RETURN C(2*y);
        else RETURN A(y-1);
    endif;
    End A;
    PROC D(x: int);
    Begin
        Print("Answer is: %d
",x+k);
    End D;
    /* Body of MAIN begins here. */
    Begin
        k := 2;
        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., for 21 on the problem and 81 on the set)