

CS3723 HOMEWORK #6, DUE: 2pm, 3/25/08 3/27/08 (Tuesday Thursday after Spring Break)

Note: If you collaborated with your classmates or used their notes, please note which classmates you collaborated with. If you use an external source, besides the text book, lectures, notes provided by the instructor, and your own intellect, please cite that source. Use quote marks if you are quoting material word-for-word from any source (including the text book).

SCOPE, PROCEDURE CALLS, AND ACTIVATION RECORDS (Chapter 7)

- List five characteristic properties of block-structured languages?
 - What does it mean that "blocks can be nested, but cannot partially overlap?"
 - Give an example of properly nested blocks.
- A definition of *global variables* is given on p. 163 of Mitchell.
 - How does this differ from what is conventionally considered global variables in a C or C++? (cf. <http://www.cplusplus.com/doc/tutorial/variables.html>)
 - Give an example of a variable declaration in a C program that would be 'global' according to Mitchell's definition but would not normally be called 'global' by a C/C++ programmer.
 - What λ -calculus concept most closely matches Mitchell's notion of 'global' variables?

3. Consider the program:

```
1  int x = 22;
2
3  int f(int y) {
4      return x + y;
5  }
6
7  main() {
8      int x = 33;
9      {
10         int x = 44;
11         printf("%d", f(7));
12     }
13 }
```

- To which declaration would the x on line 4 refer to under static scoping rules.
 - To which declaration would the x on line 4 refer to under dynamic scoping rules.
 - What is the programs output under static scoping rules?
 - What is the programs output under dynamic scoping rules?
- List 5 languages that are (at least mostly) statically scoped.
 - List 2 languages that are dynamically
 - Why do you think most modern general-purpose programming languages use static scoping?
 - List 2 language features that are dynamically scoped in C, C++, or Java.
 - Would static scoping have made sense for these features? Why or Why not?
 - What is the difference between the scope of a variable and its lifetime?
 - Give an example of variable and a statement in a (C, ML, or Java) program for which: the variable is not in scope at the statement, but the variable is live when the statement executes.
 - For each of the following activation record fields describe what is stored in that field: (a) control link (b) access link (c) return address (d) return-result address (e) actual parameters (f) local variables (g) temporary storage

9. (a) For each of the following programming language features, list the key implementation constructs that are used to implement them: (i) blocks (ii) nested blocks and recursive functions and procedures (iii) static scoping in recursive functions and procedures (iv) functions as values (v) functions values returned from nested scope
- (b) Why are closures and access links needed in ML but not in C?
10. Do exercise 7.1 in Mitchell (p. 191).
11. Explain what the difference between *pass-by-name* (p. 96), *pass-by-value*, and *pass-by-reference* parameter passing.

ACTIVATION RECORD EXERCISES — 15pts each

12. Do exercise 7.12 in Mitchell.
13. Do exercise 7.13 in Mitchell.
14. Suppose that activation records (AR) for a C implementation are stack allocated with the form given:

control link
return-result address
return address
parameter 1
...
parameter n
local variable 1
...
local variable n
temporary 1
...
temporary n

The stack should grow towards the bottom of the page.

Based on the C program:

```

1 int fac(int n)
2 {
3     if (n <= 1) {
4         return 1;
5     } else {
6         int x = -2;
7         x = fac(n - 1);
8         return n * x;
9     }
10 }
11
12 main()
13 {
14     int val = -1;
15     val = fac(3);
16     printf("%d\n", n);
17 }
```

Draw the activation records that are on the stack just after the execution of line 8 during the call to `fac()` with the parameter 2. Use the line numbers for return addresses. Draw directed arcs for the control links. Clearly label the values of local variables and parameters. Label each activation record with the procedure name. Also show which activation record is pointed to by the environmental pointer. The activation record for `main` is shown:

