

CS 345 - Programming Languages Spring 2008

Homework #4

Due: 2pm CST (in class), March 6, 2008

YOUR NAME: _____

Collaboration policy

No collaboration is permitted on this assignment. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Sciences department code of conduct can be found at <http://www.cs.utexas.edu/users/ear/CodeOfConduct.html>

Late submission policy

This homework is due at the **beginning of class** on **March 6**. All late submissions will be subject to the following policy.

You start the semester with a credit of 3 late days. For the purpose of counting late days, a "day" is 24 hours starting at 2pm on the assignment's due date. Partial days are rounded up to the next full day. You are free to divide your late days among the take-home assignments any way you want: submit four assignments 1 day late, submit one assignment 3 days late, *etc.* After your 3 days are used up, no late submissions will be accepted and you will automatically receive 0 points for each late assignment.

You may submit late assignments to Vitaly Shmatikov (TAY 4.115C—slide under the door if the office is locked). **If you are submitting late, please indicate how many late days you are using.**

Write the number of late days you are using: _____

Homework #4 (32 points)

Tips and hints

We have created an account for each student enrolled in the course at this website: <http://z.cs.utexas.edu/users/arvindn/wordpress> (or follow the link from the course website) You can obtain useful tips for the current take-home assignment by logging into your account. Your username is your UT EID. You must **update your password** for each assignment.

We may use some of the passwords in our research on password security. **IMPORTANT:** Do **not** use your UT Direct, UT CS or any other existing password for the above account!

Problem 1

The following program is written in an Algol-like language.

```
begin
  integer i;
  procedure foo(x,y);
    integer x,y; // types of parameters
    x := x+1;
    y := x+1;
    x := y;
    i := i+1;
  end;
  i:=1;
  foo(i,i);
  print i;
```

What number will be printed by the program for each of the following parameter passing mechanisms.

Problem 1a (1 point)

Pass by value.

Problem 1b (1 point)

Pass by reference.

Problem 1c (2 points)

Pass by value-result.¹

Problem 2

Consider the following ML expression:

```
val x=1;
fun foo(y) = x+y-1;
fun bar(f) = let val x=10 in f(x) end;
let val x=5 in bar(foo) end;
```

Problem 2a (5 points)

Draw the run-time stack, closures, and code pointers after the call to `f`. Include all activation records and make sure to indicate where access links are pointing.

¹Recall that when a function `bar` with a pass-by-value-result parameter `z` is called with the actual argument `u`, the activation record of `bar` contains a location for `z`. At the time of the call, the r-value of `u` is copied into that location. Inside the body of `bar`, values can be assigned to `z`. When `bar` returns, the actual argument `u` is assigned the r-value of `z`.

Problem 2b (2 points)

What is the value of this expression? Why?

Problem 3

Consider the following ML implementation of factorial.

```
fun fact(n) =  
  let factBody(n, base) =  
    if n=0 then base(1)  
    else let tail(i)=base(i*n)  
        in  
          factBody(n-1,tail)  
        end  
  in  
    factBody(n, fn x => x)  
  end
```

Observe that `factBody` is tail-recursive.

Problem 3a (3 points)

Fill in the following activation records resulting from the execution of `fact(2)`. Assume that no optimizations are done. You may need to draw closures and/or other data.

factBody(2, fn $x \Rightarrow x$)	access link	
	n	
	base	
factBody(1, tail)	access link	
	n	
	base	
factBody(0, tail)	access link	
	n	
	base	

Problem 3b (2 points)

Explain why this function is more difficult to optimize than the tail-recursive functions we discussed in class.

Problem 4

Determine the ML type for each of the following declarations. Feel free to type the declarations into an ML compiler (just run `sml` on any UTCS machine) to determine the type, but make sure to explain in a couple of sentences why the type is what it is.

Problem 4a (2 points)

```
fun a(x,y) = x + y/2.0;
```

Problem 4b (2 points)

```
fun b(f) = fn x => f(x+1);
```

Problem 4c (2 points)

```
fun c(f,g,x) = f(g(x));
```

Problem 4d (2 points)

```
fun addToList(nil, x) = x
|  addToList(x, h::l) = h::addToList(x,l);
```

Problem 4e (2 points)

The `addToList` function above has a bug. Can the type inferred for this function help the programmer notice that the function is implemented incorrectly? How?

Problem 5

Recall that we defined *garbage* to be any memory area which is not reachable from one of the root locations. Let's call this Definition A.

Another way to define garbage is the following Definition B: *At any point in the execution of the program, a memory location is garbage is no continued execution of the program from this point can access this location.*

Problem 5a (2 points)

If a memory location is garbage according to Definition A, must it also be garbage according to Definition B? Explain.

Problem 5b (2 points)

If a memory location is garbage according to Definition B, must it also be garbage according to Definition A? Explain.

Problem 5c (2 points)

Is it possible to design a garbage collector that would collect everything that is garbage according to Definition B? Explain.