CS 345 - Programming Languages
Fall 2010

Homework #3

Due: 2pm CDT (in class), October 19, 2010

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 Science Department Code of Conduct can be found at `http://www.cs.utexas.edu/academics/conduct/`.

Late submission policy

This homework is due at the **beginning of class** on **October 19**. 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 (CSA 1.114—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 #3 (35 points)

Problem 1

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 1a (2 points)

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

Problem 1b (2 points)

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

Problem 1c (2 points)

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

Problem 2

Consider the following ML expression:

```
val y=2;
fun f(x) = x*y;
fun g(h) = let val y=5 in 3+h(y) end;
let val y=3 in g(f) end;
```

Problem 2a (5 points)

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

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 (4 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 => x)	access link	
	n	
	base	
factBody(1, tail)	access link	
	n	
	base	
factBody(0, tail)	access link	
	n	
	base	

Problem 3b (3 points)

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

Problem 5 (5 points)

Here is a JavaScript function that uses an exception called `OddExcpt` (if you haven't seen JavaScript before, don't worry, the meaning of this code should be obvious).

```
function OddExcpt(){
   this.desc="Odd exception";
}

function f(n) {
  if (n==0)
     return 1;
  if (n==1)
     throw new OddExcpt;
  if (n==3)
     return f(3-2);
  try{
     return f(n-2); }
  catch(e){ return -n; }
}
```

When `f(11)` is executed, the following steps will be performed:

```
call f(11)
call f(9)
call f(7)
...
```

Write down the remaining steps that will be executed. Include only the following:

- function call (with argument)
- function return (with return value)
- raise an exception
- pop activation record of function off stack without returning control to the function
- handle an exception

Assume that if f calls g and g raises an exception that f does not handle, then the activation record of f is popped off the stack without returning control to the function f.

Problem 5

Determine the ML type for each of the following declarations. Feel free to type the declarations into an ML interpreter (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 5a (2 points)

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

Problem 5b (2 points)

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

Problem 5c (2 points)

```
fun c(w, x, y, z) = if w(x) then x(y) else z;
```

Problem 5d (2 points)

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

Problem 5e (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?