CS 421 Spring 2013 Midterm 2

Monday, April 1, 2013

Name	
NetID	

- You have **70 minutes** to complete this exam
- This is a **closed book** exam.
- Do not share anything with other students. Do not talk to other students. Do not look at another student's exam. Do not expose your exam to easy viewing by other students. Violation of any of these rules will count as cheating.
- If you believe there is an error, or an ambiguous question, seek clarification from one of the TAs. You must use a whisper, or write your question out.
- Including this cover sheet, there are 7 pages to the exam. Please verify that you have all 7 pages.
- Please write your name and NetID in the spaces above, and at the top of every page.

Question	**Value**	**Score**
1	20	
2	25	
3	15	
4	20	
5	10	
6	10	
Total	**100**	

1. (20 pts) Fill in the blanks:

   (a) The three stages of the back-end of a compiler are, in order:

      i. translation from AST to ________________________________

      ii. machine-independent ________________________________

      iii. code ________________________________

   (b) A major disadvantage of reference-counting is its inability to handle ____________________ data structures.

   (c) Mark-and-sweep garbage collection takes time proportional to the size of ____________________, while stop-and-copy g.c. takes time proportional to the size of the ____________________ data.

   (d) Normally, the statements "`x = f(0);`" and "`{ x = f(0); x = f(0); }`" are equivalent. However, they may give different results if `f` has ____________________.

   (e) If `A` is declared in C as `int[5][10]`, and integers are four bytes, then `A[i][j]` is at location address(A) + ____________________.

   (f) Programs can be verified in a proof system whose judgments have the form "$P\,\{\,A\,\}\,Q$". These judgments are called ____________________. $P\,\{\,A\,\}\,Q$ means if P is true, then Q will be true after A finishes executing. However, it does not say that A actually finishes executing; A could go into an infinite loop (and make the judgment vacuously true). For this reason, $P\,\{\,A\,\}\,Q$ is said to express the "____________________ correctness" of A.

2. (25 pts) In this question, we will ask you to fill in SOS rules in the style of MP6 (MiniJava interpretation with one-level store), MP7 (two-level store), and MP8 (compilation). As a reminder, the judgments in these three systems are:

	Expressions	**Statements**
One-level store (MP6)	$e, \sigma, \pi \Downarrow v$	$S, \sigma, \pi \Rightarrow \sigma'$
Two-level store (MP7)	$e, (\rho, \eta), \pi \Downarrow v, \eta'$	$S, (\rho, \eta), \pi \Rightarrow \rho', \eta'$
Compilation (MP8)	$e, loc \rightsquigarrow code$	$S, m \rightsquigarrow code, m'$

Specifically, for each of the three assignments, we will ask you to give rules for the `if` statement, which you implemented in that assignment, and for `while`, which you didn't. For `if` in MP6 and MP7, you may recall that there are two rules, one for the false case and one for the true case; for MP8, there is just one compilation rule. `while` is similar: for interpretation (MP6 and 7), there are two rules, one for when the condition is true and one for when it is false. The rules for `while` are also different in that they include a recursive execution of the `while` statement itself. We have filled in that line for you. Again, for compilation, there is just one rule (and it is not recursive). In all three parts of this problem, you need to fill in the blank lines.

(a) (6 pts) One-level store:

`if` (e) $S1$ `else` $S2$, σ, $\pi \Rightarrow \sigma'$

____

____

`if` (e) $S1$ `else` $S2$, σ, $\pi \Rightarrow \sigma'$

____

____

`while` (e) S, σ, $\pi \Rightarrow \sigma''$

____

____

`while` (e) S, σ', $\pi \Rightarrow \sigma''$

`while` (e) S, σ, $\pi \Rightarrow \sigma$

____

(b) (6 pts) Two-level store:

`if (`e`)` $S1$ `else` $S2$, (ρ, η), $\pi \Rightarrow \hat{\rho}$, $\hat{\eta}$

____

____

`if (`e`)` $S1$ `else` $S2$, (ρ, η), $\pi \Rightarrow \hat{\rho}$, $\hat{\eta}$

____

____

`while (`e`)` S, (ρ, η), $\pi \Rightarrow \hat{\rho}$, $\hat{\eta}$

____

____

`while (`e`)` S, __________, $\pi \Rightarrow \hat{\rho}$, $\hat{\eta}$

`while (`e`)` S, (ρ, η), $\pi \Rightarrow \eta'$

____

(c) (7 pts) Compilation

`if (`e`)` $S1$ `else` $S2$, $m \rightsquigarrow$ il @ [CJUMP loc, __________, __________] @ il1

@ [JUMP __________] @ il2 , m''

e, loc $\rightsquigarrow$ il

$S1$, __________ $\rightsquigarrow$ il1, __________

$S2$, __________ $\rightsquigarrow$ il2, __________

`while (`e`)` S, $m \rightsquigarrow$ [JUMP ________] @ il2 @ il1 @ [CJUMP loc, ________, ________],

e, loc $\rightsquigarrow$ il1

____

3. (15 pts) These questions concern the substitution model for evaluation of MiniOCaml (MP 9).

   (a) (6 pts) In this model, some expressions are designated as “values,” which means that they are considered to be fully evaluated and need no more simplification. Mark each of the following expressions as V for value or NV for not value:

   _______ `3`

   _______ `3+4`

   _______ `[3+4; 5]`

   _______ `fun x -> 3+4`

   _______ `fun x -> let y=1 in y+2`

   _______ `(fun x -> f 4)(fun x -> x+x)`

   (b) (14 pts) Some of the SOS rules for the substitution model for MiniOCaml are given at the bottom of the next question (just to remind you what they look like). Give SOS rules for two expressions (where you may assume x and y are different):

   - `let` x `=` e `and` y `=` e' `in` e'' evaluates e and e' in the same environment, and then evaluates e'' with x and y bound to their values. (This is how `let` with multiple declarations works in OCaml.)
   - `let*` x `=` e `and` y `=` e' `in` e'' is “sequential let;” it evaluates e as above, but then evaluates e' in an environment in which x has its new value; then it evaluates e'' as before.

   $$\texttt{let}\ x\ \texttt{=}\ e\ \texttt{and}\ y\ \texttt{=}\ e'\ \texttt{in}\ e'' \Downarrow v$$

   $$\texttt{let*}\ x\ \texttt{=}\ e\ \texttt{and}\ y\ \texttt{=}\ e'\ \texttt{in}\ e'' \Downarrow v$$

4. (20 pts) Give the complete evaluation tree for the following MiniOCaml expression, using the substitution model. The rules you will need are given below. To save writing, you may refer to the expression denoted e_0 by just writing e_0. Finally, note that we have filled in one expression in this evaluation — which happens to be the longest expression that shows up. The lengths of the blank lines are not related to the sizes of the expressions that go there.

   Let e_0 be `if x=0 then fun f -> f 10 else fun f -> f x`.

   `((fun x ->` e_0`) 0 ) (fun x -> x+x)` $\Downarrow$ 20

   `if 0=0 then fun f -> f 10 else fun f -> f x` $\Downarrow$ ________________

(Const) Const x $\Downarrow$ Const x

(Fun) Fun(a,e) $\Downarrow$ Fun(a,e)

(δ) $e + e' \Downarrow v + v'$
$e \Downarrow v$
$e' \Downarrow v'$

(δ) $e = e' \Downarrow v = v'$
$e \Downarrow v$
$e' \Downarrow v'$

(If) If(e_1, e_2, e_3) $\Downarrow v$
$e_1 \Downarrow$ True
$e_2 \Downarrow v$

(App) $e\ e' \Downarrow v$
$e \Downarrow$ Fun(a, e'')
$e' \Downarrow v'$
$e''[v'/a] \Downarrow v$

5. (10 pts) Give the loop invariant of this loop; mathematical notation is best, but you can use English as long as your answer is clear and precise. Then give a termination function.

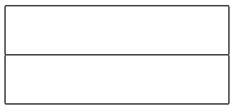
```
max = 0 ∧ i = 1 ∧ n > 1 {
   while (i < n) {
        if (a[i] > a[max])
           max = a[i];
        i = i+1;
   }
} ∀0 ≤ j < n.a[max] ≥ a[j]
```

Invariant:

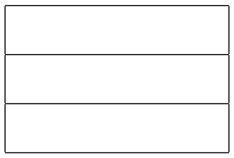
Termination function: T(max, i, n) =

6. (10 pts) Fill in the v-tables for these classes, by drawing arrows from each entry in the v-table to one of the function definitions (as we did in class).

```
class C {
    f () {...}
    g () {...}
}
```



```
class D extends C {
    h () {...}
    f () {...}
}
```



```
class E extends D {
    i () {...}
    g () {...}
}
```

```
class F extends E {
    i () {...}
    g () {...}
}
```