CHALMERS TEKNISKA HÖGSKOLA
Dept. of Computer Science and Engineering
John Hughes

Tuesday ?th April, 2010.
Programming Paradigms
DAT120(CTH) / DIT330(GU)

Exam in Programming Paradigms

Tuesday 16th December, 2007, EM.
Lecturer: John Hughes, tel 070 756 3760.

Permitted aids:
English-Swedish or English-other language dictionary.

There are five questions, one on each paradigm, worth 12 points each for a total of 60 points. 24 points is required to pass (grade 3), 36 points is required for grade 4, and 48 points is required for grade 5.

Imperative Programming [12 points]

In this part of the exam, we use the following notation in expressions:

- `* q` means: memory cell whose adress is `q`. (Note that `q` must be an address)
- `& a` means: address of `a`. (Note that `a` must be an l-value)

- Adresses and L-values **[4 points]**

  Which of the following these expressions are l-values? Which expressions are pointers? (`a`, `b` denote integers variables; `p`, `q` denote variables containing pointers to integers.)

  Reproduce the following table and replace the question marks with "yes" or "no" appropriately.

expression	l-value	pointer
a	?	?
p	?	?
& a	?	?
& p	?	?
* (& a)	?	?
* (& p)	?	?

- Parameter passing

  Consider the following program.

```
f (a, b : integers passed by value-result) {
    a := b
    b := b - 3;
    return;
}

i: integer
x: array of integers
i := 1;
x[0] := 3;
x[1] := 4;
f(i,x[0]);
print (i + x[0] + x[1]);
```

  What is printed? **[2 points]**

  Translation to call-by-reference. **[6 points]**

  Translate the function `f` *and* its call to a language that does not support call by value-result, but only call by reference. (In particular the function may not return any value via a "return" instruction). Do so by using temporary variables. You are not allowed to change anything else. In particular, the "algorithm" and the declatations of `x` and `i` must remain the same. The value printed must remain the same. Assume copy semantics for assignment and early l-value (memory location) computation.

Object-Oriented Programming [12 points]

- Subtyping.

  1. State the substitution principle of Liskov. **[3 points]**
  2. Is subtyping a transitive relation (A :< B and B :< C implies that A :< C)? **[1 point]**
     Justify by using the substitution principle. **[2 points]**

- Algebraic specification.

  Consider the following specification for the sort `S`.

```
Signature:

yes : S
no : S
maybe : S
and : S × S → S
or : S × S → S
not : S → S

Axioms (x, y are variables)

not(yes) = no
not(no) = yes
not(maybe) = maybe
or (no,x) = x
or (x,y) = or (y,x)
and (x,y) = not (or (not(x),not(y)))
```

  Assuming initial algebra semantics, which of the propositions are true? **[6 points]**

  (Copy each line and write "true" or "false" after it)

  - and(maybe,maybe) = maybe
  - or (maybe,no) = not(maybe)
  - and (yes,no) = or(maybe,no)
  - and (maybe,yes) = not(maybe)
  - and (no,yes) = or(no,no)
  - or(maybe,yes) = or(yes,maybe)