CPSC 320 Winter 2001
Midterm II—Wednesday, 14 February 2001

Name(Printed) : _____

Signature : _____

StudentNumber : _____

ACRE	AREA	BALE	BAND	BARD	BASS
BETA	BIRD	BLOT	BOOK	BREW	CAMP
CHIP	CLAN	COAT	COIL	CORN	CROW
CURL	DARK	DEER	DOSE	DROP	DUCK
DUSK	FARE	FILM	FLAX	GAZE	GIFT
GOLD	GULF	HINT	HORN	HULL	IBOU
INCH	IRIS	ISLE	KERN	KILN	KITE
LANE	LARK	LENS	LOFT	LURE	MALT
MANX	MESH	MINK	MOTH	MOVE	MUSK
NAVY	NEWT	NOON	OATS	OBOE	OPAL
PARK	PINE	POET	RAFT	REED	RING
RUBY	RUFF	SEAM	SEED	SHOP	SILK
SINE	SNIP	SOAP	STUB	TASK	TAXI
TEAM	TELL	TEXT	TIDE	TILT	TOIL
TOME	TOUR	TURN	VANE	VISA	WALL
WICK	WOLF	WRIT	YARD		

Question	Score
1	/4
2	/4
3	/8
4	/3
5	/4
6	/2
7	/2
8	/2
9	/3
10	/2
11	/9
12	/7
Total	/50

Instructions

- Write the word marked above on each page of questions of your exam. Do not put any other identifying marks on any page of your exam. Failure to put the circled word on a page of your exam may result in no marks being awarded for that page.
- *Read each question carefully. Ask yourself what the point of the question is.* Answer each question. *Check to make sure that you have answered the question asked.*
- Answer all questions on the exam sheet. If you do some of your work on the back of a page, clearly indicate to the marker what work corresponds with which question.
- This is a **50** minute exam. This exam contains **7** pages of questions not including this cover page. Make sure that you have all of them.
- Partial marks shall be awarded for clearly identified work.
- This exam counts as **20%** of your total grade.

The Semantics of While loops

1. For brevity let $f = \mathcal{C}[\![\texttt{while } c \texttt{ do } p \texttt{ endwhile}]\!]$. We derived that f must satisfy the equation

$$f(\sigma) = \begin{cases} \big(f \diamond \mathcal{C}[\![p]\!]\big)(\sigma) & \text{if } \mathcal{E}[\![c]\!](\sigma) > 0 \\ \sigma & \text{otherwise} \end{cases} \tag{1}$$

(2) (a) Explain roughly how we got the right hand side of (1).

(2) (b) Equation (1) sometimes has more than one solution. What relation does the correct solution have to other solutions?

(1) **2.** (a) What does $\perp$ symbolize?

(1) (b) We used to use $\mathcal{C}[\![s_1\,;s_2]\!] = \mathcal{C}[\![s_2]\!] \circ \mathcal{C}[\![s_1]\!]$. With `while`-loops in our language we must write $\mathcal{C}[\![s_1\,;s_2]\!] = \mathcal{C}[\![s_2]\!] \diamond \mathcal{C}[\![s_1]\!]$. What is new in our model of the execution of s_1 that means that ordinary function composition is not appropriate here?

(2) (c) Assuming that f and g are functions from $\mathcal{S}$ to $\mathcal{S}_\perp$, give the definition

of $[f \diamond g](\sigma)$.

(8) **3.** Match the following programs with their denotations. Note that we are assuming a language where a real number r is considered to be `true` as a boolean value if $r > 0$.

A
```
while -5.0
do
   x := 0.0 - y ;
end while
```

B
```
while 5
do
    x := 0.0 - y ;
end while
```

C
```
while y
do
    x := 0.0 - y ;
end while
```

D
```
while x * y
do
    x := x - y ;
    y := x + y ;
    x := x - y ;
end while
```

(a) $\sigma \mapsto \bot$

(b) $\sigma \mapsto \sigma$

(c) $\sigma \mapsto \begin{cases} \sigma & \text{if } \sigma(\mathbf{y})\sigma(\mathbf{y}) \leq 0 \\ \sigma[\mathbf{x}/(-\sigma(\mathbf{y}))] & \\ \ [\mathbf{y}/\sigma(\mathbf{x})] & \text{otherwise.} \end{cases}$

(d) $\sigma \mapsto \begin{cases} \sigma & \text{if } \sigma(\mathbf{y}) \leq 0 \\ \bot & \text{otherwise} \end{cases}$.

(e) $\sigma \mapsto \begin{cases} \bot & \text{if } \sigma(\mathbf{y}) \leq 0 \\ \sigma & \text{otherwise} \end{cases}$.

(f) $\sigma \mapsto \begin{cases} \sigma & \text{if } \sigma(\mathbf{x})\sigma(\mathbf{y}) > 0 \\ \sigma[\mathbf{x}/(-\sigma(\mathbf{y}))] & \text{otherwise.} \end{cases}$

A. matches ____. **B.** matches ____. **C.** matches ____.

D. matches ____.

Data types

(1) **4.** (a) What do we mean by a *first-class* value?

(2) (b) Give an example of a language, and a kind of value in that language that is not first-class, and the evidence you have for the fact that it is not first-class.

5. Louden gives ***Definition 1.*** A **data type** is a set of values. as his first definition of data type.

(2) (a) Why is this definition too broad?

(2) (b) How can it be made more precise?

(2) **6.** *(Circle the best choice.)* The C++ type

```
struct T1 { double z ; int w[2] ; } ;
```

considered as a set most closely corresponds to

(a) $\mathbb{R} \times \mathbb{Z}^2$
(b) $\mathbb{R} \cup \mathbb{Z}^2$
(c) $\mathbb{R} \to \mathbb{Z}^2$
(d) `T1` considered as a set in no way corresponds to any of the above.

(2) **7.** What are some properties that help define a character type, or distinguish one character type from another?

(2) **8.** Many Object-Oriented programming languages don't have explicit subset type constructors, yet they allow one to model that type Y is a subset of type X. What is the connection between subsets and object orientation?

True or False

1 each **9.** Circle **TRUE** or **FALSE** as appropriate. Questions that don't clearly indicate *one* choice shall be marked wrong.

(a) PASCAL has a explicit powerset type constructor. **TRUE** **FALSE**

(b) C++ has an explicit subset type constructor. **TRUE** **FALSE**

(c) Union types often have a syntactically attached tag or discriminator. **TRUE** **FALSE**

(d) The C++ `union` types have a syntactically attached tag or discriminator. **TRUE** **FALSE**

(e) In PASCAL strong-typing is weakened somewhat by the ability of a programmer to arbitrarily manipulate the value of the tag in a record variant. **TRUE** **FALSE**

(f) In ADA strong-typing is weakened somewhat by the ability of a programmer to arbitrarily manipulate the value of the tag in a record variant. **TRUE** **FALSE**

(g) Pointers are a form of set-theoretic type constructor. **TRUE** **FALSE**

Type equivalence

(3) **10.** Name three common forms of type equivalence.

11. In Standard ML the `datatype` keyword introduces a new (union-like) type. For instance, in lecture, I gave an example something like

```
datatype number = R of real | I of Int ;
fun n2real (R r) = r
  | n2real (I i) = Real.fromInt i ;
```

If we next define another data type by

```
datatype number2 = R of real | I of Int ;
```

and try computing `n2real (I 5)`, we'll get a type mismatch because Standard ML considers `(I 5)` to be of type `number2` and doesn't consider `number` and `number2` to be equivalent types.

(1) (a) In this case, what form of type equivalence is Standard ML *not* using in comparing `number` and `number2`?

(1) (b) Standard ML has another sytax for datatype statements that looks like

```
datatype number3 = datatype number ;
```

After such a declaration, values of type `number3` can be used where values of type `number` are expected. In this case, what form of type equivalence is Standard ML *not* using in comparing `number` and `number3`?

Type inference

12. The function

```
ostream&
print_hex_digit(ostream& out, int n)
{
    return
        (n>9)
        ? out << "ABCDEF"[n-10]
        : out << n ;
}
```

prints '`A`' when called with `print_hex_digit(cout, 10)`.

On the other hand, the function

```
ostream&
print_hexed_digit(ostream& out, int n)
{
    return
        out << ((n>9) ? "ABCDEF"[n-10] : n) ;
}
```

prints '`65`' when called with `print_hexed_digit(cout, 10)`.

The following questions relate to explaining why this happens in terms of how C++ performs type inference and its willingness to perform implicit casts.

(2) (a) Draw a parse tree or abstract syntax tree of the expression following `return` in `print_hexed_digit`.

(2) (b) Explain in general terms how C++ infers the types of the expressions.

(2) (c) Label the nodes of the tree from 0a with the types that compiler infers.

(1) (d) What implicit cast is inserted?

(2) (e) Briefly explain why the first version works.