CSE 428, Spring 1998

Midterm #1

23 February 1998

The exam consists of 5 exercises, totalling 80 points. Write your answers directly on these pages. **Please remember: write your name and SSN at the top of each sheet!**

Exercise 1 (20 pts)

Consider the following grammar for numerical expressions:

$$\begin{array}{rcl} Exp & ::= & Exp + Term \mid Term \\ Term & ::= & Fact * Term \mid Fact \\ Fact & ::= & -Num \mid (Exp) \mid Num \end{array}$$

where the nonterminals are Exp, $Term$, $Fact$, and Num; the initial symbol is Exp; and Num generates the language of natural numbers (via the usual productions, not shown here).

1. Is the grammar ambiguous? (yes or not)
2. Which kind of associativity is enforced for $+$? And for $*$?
3. For each of the following strings, say whether it is generated by the above grammar or not
   (a) $5 + -2$
   (b) $5 + (-2)$
   (c) $5 - 2$
   (d) $-5 * (3 + 2)$
   (e) $-(5 * 3) + 2$

   Replace the production $Fact ::= -Num$ with $Fact ::= -Fact$. Is the new grammar ambiguous? Does it generate the same language?

Exercise 2 (10 pts)

Consider the following expression

$$\textbf{let } x = 1 \textbf{ in } x < y \textbf{ endlet}$$

What type (if any) will be associated to this expression by the type checker starting from:

1. The empty static environment
2. A static environment which contains the association between the identifier y and the type **int**
3. A static environment which contains the association between the identifiers y and x and the types **int** and **bool** respectively

Exercise 3 (20 pts)

The "Fibonacci" function is defined ricorsively as:

$$\begin{aligned} fibo(0) &= 0 \\ fibo(1) &= 1 \\ fibo(x+2) &= fibo(x) + fibo(x+1) \end{aligned}$$

Write an iterative implementation of this function, i.e. write a command which takes in input an integer k and outputs $fibo(k)$ if $k \geq 0$, and 0 otherwise. The command **must** be written by using only the constructs of the mini-language, i.e. **while** (or **for**), assignment, block, and input-output commands. In particular, no recursive procedure or function should be used.

Exercise 4 (10 pts)

Consider the following command:

> **begin**
> $p, q :\uparrow$ **int**
> **in**
> **new**(p);
> $q := p$;
> $p \uparrow := 1$;
> $q \uparrow := p \uparrow +1$;
> **print**$(p \uparrow)$;
> **dispose**(p);
> **print**$(q \uparrow)$
> **end**

1. What is the value printed by **print**$(p \uparrow)$?
2. Describe graphically the situation (environment and state) after the execution of **dispose**(p)
3. Discuss what can happen when **print**$(q \uparrow)$ is executed.

Exercise 5 (20 pts)

Consider the following three possible definitions of the procedure *foo*

Call by value: **procedure** $foo(x, y, z : \textbf{int}); C$

Call by reference: $\quad$ **procedure** $foo(\mathbf{var}\ x, y, z : \mathbf{int}); C$

Call by value-result: $\quad$ **procedure** $foo(\mathbf{value-result}\ x, y, z : \mathbf{int}); C$

(Note: we are using the Pascal notation: the keywords **var** and **value-result** refer to all the three identifiers.) In all the above declarations, C is the command

> **begin**
> $y := y + z;$
> $x := z$
> **end**

Consider now the following fragment of program:

> **begin**
> $x, w : int$
> **in**
> $x := 0;$
> $w := 1;$
> $foo(x, w, w);$
> $\mathbf{print}(x)$
> **end**

1. Describe graphically the environment immediately after the activation of *foo*, in all three cases.

2. What is the effect of the execution of the command $\mathbf{print}(x)$, in the three cases?