CSE 428, Spring 1999. Final exam, 5 May 1999

The exam consists of 9 exercises, totalling 100 points. Write your answers directly on these pages. **Please remember to write your name and SSN at the top of each sheet.**

Exercise 1 (15 pts)

Consider the following grammar, defining a fragment of ML:

```
Exp ::= Ide                  (identifier)
      | Exp Exp              (application)
      | fn Ide => Exp        (abstraction)
      | (Exp)

Ide ::= x | y | z
```

1. (5 pts) Is this grammar ambiguous? Justify your answer.

2. (10 pts) Define an equivalent grammar in which application is left associative, and it has priority w.r.t. abstraction.

Exercise 2 (9 pts)

Consider the following sequence of declarations in a Pascal-like language:

```
var z: integer;
procedure p (mode x, y : integer)
   begin x := x + 1; y := y + 1; z := z + 3 end;
```

where `mode` represents one of the following parameter-passing modes: value, reference, or value-result.

Say what is the value printed by the following fragment of code

```
z := 1;
p(z,z);
print(z);
```

In case of

1. (3 pts) pass by value.

2. (3 pts) pass by reference.

3. (3 pts) pass by value-result.

Exercise 3 (9 pts)

For each of the following procedure declarations in C++, say whether its execution leaves dangling pointers and/or memory leaks. Assume that `p` is a global variable declared as a pointer to `int` and initialized to `NULL`.

1. (3 pts) `void foo() { int x; p = &x; }`

2. (3 pts) `void foo() { int *q; q = new int; p = q; }`

3. (3 pts) `void foo() { int *q; q = new int; p = q; delete q; }`

Exercise 4 (10 pts)

Consider the following class declarations in C++:

```
class C { protected: int x; ...};
class C1: public C { private: int x1; ...};
class C2: public C1 { private: int x2; ...};
class C3: public C { protected: int x3; ...};
```

1. (2 pts) Draw a tree-like structure illustrating the class hierarchy

2. For each of the following instructions, say whether it is allowed in the body of `C2` or not

   (a) (2 pts) `x = x+1;`

   (b) (2 pts) `x1 = x1+1;`

   (c) (2 pts) `x2 = x2+1;`

   (d) (2 pts) `x3 = x3+1;`

Exercise 5 (14 pts)

The following declaration in Java defines a data structure "Queue" suitable for a single-thread use.

```
class Queue

{

    private class Element

   {
        int info;

        Element next;

        Element(int n) { info = n; next = null; }
    }

    private Element first;

    private Element last;

    Queue() { first = null; }

    public boolean is_empty()

    {
       return (first == null);
    }

    public void insert(int n)

    {
       if (first == null)

         { first = last = new Element(n); }

       else

         { last = last.next = new Element(n); }
    }

    public int pop()

    {
        int n = first.info;

        first = first.next;

        return n;
    }
}
```

Modify the above declaration so to make it suitable for a multiple-threads use (i.e. we want the queue to be shared by many threads). Write the needed additional code in the empty lines, and delete the existing code where necessary. **Note:** In the code above, when used in a single-thread context, we assume that the caller of the `pop()` method checks first that `is_empty()` is false. You can make the same assumption for the multiple-threads use if you want, but beware: it may not be sufficient.

Exercise 6 (10 pts)

Consider the following datatype declaration in ML:

```
datatype 'a bintree = leaf of 'a | node of 'a bintree * 'a bintree
```

(note that, with this declaration, we have information only on the leaves, and the empty tree is not represented). Define a function `sum_bintree: int bintree -> int` such that `sum_bintree(t)` returns the sum of all the numbers contained in `t`.

Exercise 7 (10 pts)

Define a function `map_fun_list: ('a -> 'b) list -> 'a -> 'b list` such that `map_fun_list [f_1,f_2,...,f_n] x = [f_1(x),f_2(x),...,f_n(x)]`. Example: If `fact` is defined as the factorial function, and `power_2` as the function which, given n, returns 2^n, then `map_fun_list [fact,power_2] 3` should give as result the list `[6,8]`.

Exercise 8 (9 pts)

For each of the following ML expressions, say whether it is typeable, and, in the affirmative case, give is its most general type

1. (3 pts) `fn f => fn x => f x x;`

2. (3 pts) `fn f => fn x => f (x x);`

3. (3 pts) `fn f => fn x => f (x,x);`

Exercise 9 (14 pts)

Define a Prolog predicate `delete_repetitions(L,K)` which holds iff `K` is the list obtained from `L` by deleting all the repeated occurrences of the same element. The definition should work correctly at least with queries in which `L` is fully instantiated. Example:

```
?- delete_repetitions([a,b,a,b,b,c],K).
K = [a,b,c];
no
```

If you use auxiliary predicates, please define them and explain in words what's their intended meaning. You are free to choose which occurrences to delete (i.e. you can leave the first occurrence, or the last, or one in the middle. . . whatever you find more convenient).