CSE 428, Spring 2001. Final exam, 4 May 2001

The exam consists of 6 questions, totalling 100 points. Write your answers directly on these pages. **Please remember to write your name and student ID at the top of each sheet.**

Question 1 (total 15 pts)

Consider the following grammar, defining a small language of expressions:

```
Exp ::= Num | Exp ^ Exp | Num * Exp | (Exp)
Num ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

where `Exp` and `Num` are non-terminals and all the other symbols are terminal. `Exp` is the start symbol.

1. **(4 pts)** For each of the following strings, say whether it is derivable from the grammar or not
   (a) `25 * 3`
   (b) `5 ^ 2 * 3`
   (c) `(5 ^ 2) * 3`
   (d) `5 * 2 * 3`

2. **(4 pts)** Show all the kinds of ambiguity in this grammar. For each kind of ambiguity, show two different derivation trees for the same string.

3. **(7 pts)** Give an equivalent unambiguous grammar for the same language, defined in such a way that ^ is right associative. You can choose the rule you want for the priority between $*$ and ^. (In this particular exercise, it is easier to write a grammar which gives priority to $*$.)

Question 2 (total 15 pts)

Consider the following C++ code:

```
class list { //Implementation of a list (only the methods relevant for this exercise)

private:
   class element {

   public:
      int info;
      element* next;

      element(int x, element* e) { info = x; next = e; }
   }

   element* first;

public:
   list() { first = NULL; }

   void insert(int x) { //Insert x at the beginning of the list
       first = new element(x,first);
   }

};

void add_to_list(int k, int n, list* L) { //Add n - k + 1 elements to the list
   if (k>n)
      cout << "Done \n";
   else {
      L->insert(k);
      add_to_list(k, n-1, L);
   }
}

/////////////////////////////////////////////////////////////////////////////////

int main() {
   list* L = new list();
   add_to_list(3, 5, L);
   // ...
}
```

Illustrate with a drawing the situation in memory (stack and heap) at the moment in which the instruction `cout << "Done \n";` is executed. Please make explicit all the informations stored in the activation records and in the (allocated part of the) heap at that moment. Use arrows for pointers. Remember that in C++ there is no static link (there is no need for it since nested procedure declarations are not allowed). For simplicity, you can represent in the stack only the activation records of `main()` and of `add_to_list(int,int,list*)`. You can also omit the control link and the temporaries.

Question 3 (total 20 pts)

Consider the following Java code defining the class of a bank account, and the use of this account by a class of threads `AccountHolder`. We assume that more than one AccountHolders can share the same account.

```
class Account {

   private int balance;

   public Account(int initialDeposit) { balance = initialDeposit; }

   public int getBalance() {
      return balance;
   }

   public void deposit(int amount) {
      if (amount < 0)
         error(1) // the amount to deposit should be non negative
      else
         balance = balance + amount;
   }

   public void withdraw(int amount) {
      if (amount < 0)
         error(1) // the amount to withdrow should be non negative
      else if (balance - amount < 0)
         error(2) // the balance should not become negative
      else
         balance = balance - amount;
   }
}

class AccountHolder extends Thread {
    Account a;
    ...
    public void run() {
            ...
            a.deposit(x);
            ...
            while (a.getBalance() < y) { ; } // ``;'' is the empty statement
            a.withdraw(y);
            ...
    }
}
```

This code has two problems: race conditions and busy waiting. Rewrite the code so to eliminate these two problems. Please make only the necessary modifications. You can omit the treatment of Interrupted Exceptions.

Question 4 (total 15 pts)

Consider the following ML datatype declaration for binary trees:

```
datatype 'a btree = leaf of 'a | node of ('a btree * 'a * 'a btree);
```

Define in ML a function `heights : 'a btree -> int btree` which, given a binary tree `t : 'a btree`, gives as result a tree of type `int btree` with the same structure as `t`, and with the nodes labeled by their level. You can use auxiliary functions.

For instance, on the input tree `t` represented below, `heights` should give `t'` as result.

```
        t                                         t'

        a                                         5
       / \                                       / \
      /   \                                     /   \
     b     c              heights              4     3
    / \   / \            ---------\           / \   / \
   d   e j   k           ---------/          3   1 1   2
  / \       / \                             / \       / \
 f   g     l   m                           1   2     1   1
    / \                                       / \
   h   i                                     1   1
```

Note: Please write the solution in ML. Solutions in other languages will not be considered.

Question 5 (total 20 pts)

Define in Prolog a predicate `divisor_and_rest(M,N,X,Y)` which, given two positive natural numbers `M` and `N`, returns all the pairs `X` and `Y` such that

```
N = M * X + Y
```

For instance, we should have:

```
?- divisor_and_rest(2,5,X,Y).

X = 2
Y = 1 ;

X = 1
Y = 3 ;

X = 0
Y = 5 ;

No

?- divisor_and_rest(4,15,X,Y).

X = 3
Y = 3 ;

X = 2
Y = 7 ;

X = 1
Y = 11 ;

X = 0
Y = 15 ;

No
```

The order in which the solutions are generated is not important, but please make sure that your solution generates all of them and that it does not go into an infinite loop.

Please write the solution in Prolog. Solutions in other languages will not be considered.

Question 6 (total 15 pts)

Define in Prolog a predicate `conc(L,K)` which, given a list of lists `L`, returns in `K` the list obtained by concatenating all the lists of `L`. For instance, we should have:

```
?- conc([[a],[b],[c,d]],K).

K = [a, b, c, d] ;

No

?- conc([[1],[],[1,2,3,4]],K).

K = [1, 1, 2, 3, 4] ;

No

?- conc([[[1]],[[2]]],K).

K = [[1],[2]] ;

No
```