Comparative Programming Languages 2002

MR
p595

```
\begin{verbatim}
MANIFEST {
  Num=1; Id=2; Add=3; Sub=4; Mul=5; Div=7
}

LET mknum(k) = VALOF
{ LET p = getvec(1)
  p!0, p!1 := Num, k
  RESULTIS p
}
...
LET mkdiv(x, y) = VALOF
{ LET p = getvec(2)
  p!0, p!1, p!2 := Div, x, y
  RESULTIS p
}

In C:

enum Ops {NUM, POS, NEG, ADD, SUB, MUL, DIV}

typedef struct Num Num;
typedef struct Monad Monad;
typedef struct Dyad Dyad;

typedef union Expr {
  Num *N;
  Monad *M;
  Dyad *D;
} Expr;

struct Num {
  int op;
  int val;
}

struct Monad {
  int op;
  Expr rand;
}

struct Dyad {
  int op;
  Expr left;
  Expr right;
}

Expr mknum(int n) {
  Expr res;
  Num *e = (num *) malloc(sizeof(Num));
  e->op = NUM;
  e->val = n;
  res.N = e;
  return res;
}
```

etc...

In ML:

```
datatype E = NUM of int
           | POS of E
           | NEG og E
           | ADD of E * E
           | SUB of E * E
           | MUL of E * E
           | DIV of E * E;
```

Use the constructors to make nodes, eg:

```
          ADD(NUM 1, MUL(NUM 2, NUM 3))
```

Selection is by pattern matching.