Paper 4 Question 2

AM – Compiler Construction

Solution Notes y2002p4.tex

Syntax:

```
// syntax-like things (non-regular recursion)
        ValidInput ::= Expr enter | Expr store Id enter
        Expr5 ::= ( Expr ) | Number | Id
        Expr4 ::= Expr5 ^ Expr4 | Expr5
        Expr3 ::= Expr3 * Expr4  | Expr4
        Expr2 ::= - Expr2  | abs Expr2 | Expr3
        Expr1 ::= Expr1 + Expr2  | Expr2
        Expr ::= Expr1
// lexical-like things (regular recursion)
        Letter ::= a | b | ... | z
        Id ::= Letter | Letter Idtail
        Idtail ::= Digit | Letter | Idtail Digit | Idtail Letter
        Digit ::= 0 | ... | 9
        Integer ::= Integer | Integer Digit
        Number ::= Integer | Integer e + Integer | Integer e - Integer
                 | . Integer | . Integer e + Integer | . Integer e - Integer
                 | Integer . | Integer . e + Integer | Integer . e - Integer
```

NB, say this

[Most candidates misread "at least one digit, possibly interspersed with a decimal point"]

Yacc input (with semantic actions):

```
%token NUMBER
%token ID
%token ABS
%token STORE
%%
validinput: expr '\n'  { printf("%d\n", $1); }
          | expr STORE ID '\n'  { printf("%d ST %d\n", $1, $3); } ;
expr5: '(' expr ')'      { $$ = $2; }
     | NUMBER
     | ID ;
expr4: expr5 '^' expr4   { $$ = ipow($1,$3); }
     | expr5 ;
expr3: expr3 '*' expr4   { $$ = $1 * $3; }
     | expr4 ;
expr2: '-' expr2         { $$ = - $2; }
     | ABS expr2         { $$ = $2<0 ? -$2 : $2; }
     | expr3 ;
expr1: expr1 '+' expr2   { $$ = $1 + $3; }
     | expr2 ;
expr: expr1
%%
```