```
node * parse_statement(lexan & LEX)
{ LEX.nextlex();


  if (LEX.kind == LX_RW_print)
  { node * r = new node(N_printstmt);
    r->subtree.push_back(parse_expression(LEX));
    return r; }


  else if (LEX.kind == LX_RW_when)
  { node * r = new node(N_whenstmt);
    r->subtree.push_back(parse_expression(LEX));
    r->subtree.push_back(parse_statement(LEX));
    return r; }


  else if (LEX.kind == LX_RW_var)
  { node * r = new node(N_vardecl);
    LEX.nextlex();
    if (LEX.kind != LX_variable)
      LEX.error("in assignment, expecting variable, found " + LEX.form);
    r->syminfo = LEX.syminfo;
    LEX.nextlex();
    if (LEX.kind != LX_OP_assign)
      LEX.error("in assignment, expecting =, found " + LEX.form);
    r->subtree.push_back(parse_expression(LEX));
    return r; }


  else if (LEX.kind == LX_opencurly)
  { node * r = new node(N_sequence);
    while (true)
    { r->subtree.push_back(parse_statement(LEX));
      LEX.nextlex();
      if (LEX.kind != LX_semicolon)
        break; }
    if (LEX.kind != LX_closecurly)
      LEX.error("expecting close curly, found " + LEX.form);
    return r; }


  else
    LEX.error("expecting statement, found " + LEX.form); }
```

```
$ lang4
{ var x = (1+2); var y = (x-1); when (y>x) print y; print (x*3) }
sequence
   vardecl syminfo = x @ 0x804e080
      binaryexp intvalue=3
         integer intvalue=1
         integer intvalue=2
   vardecl syminfo = y @ 0x804e0d0
      binaryexp intvalue=4
         variable syminfo = x @ 0x804e080
         integer intvalue=1
   whenstmt
      binaryexp intvalue=11
         variable syminfo = y @ 0x804e0d0
         variable syminfo = x @ 0x804e080
      printstmt
         variable syminfo = y @ 0x804e0d0
   printstmt
      binaryexp intvalue=5
         variable syminfo = x @ 0x804e080
         integer intvalue=3
```