```
# A utility for converting older style Ghilbert proofs (terms written in
# RPN) to sexp terms.

import sys
import array

from scanner import Scanner

def echo_tok_nonwhite(s, sout):
    while 1:
     tok = s.get_tok()
     if tok == None:
         return None
     if sout:
         sout.write(tok[-1])
     if tok[0] != 'white':
         return tok

# parse a sexp, echoing to sout
# return value is None, ')', or sexp
def echo_sexp(s, sout):
    while 1:
     tok = echo_tok_nonwhite(s, sout)
     if tok == None:
         return None
     if tok == '(':
         result = []
         while 1:
          subsexp = echo_sexp(s, sout)
          if subsexp == ')':
              break
          elif subsexp == None:
              raise SyntaxError('eof inside sexp')
          result.append(subsexp)
         return result
     elif tok[0] in (')', 'id'):
         return tok[-1]


def termsexp_import(fn, prefix, termmap):
    s = Scanner(file(fn))
    while 1:
     tok = s.get_tok()
     if tok == None:
         break
     if tok[0] == 'id':
         cmd = tok[1]
         if cmd == 'term':
          args = echo_sexp(s, None)
          termmap[prefix + args[1][0]] = len(args[1]) - 1
         else:
          args = echo_sexp(s, None)

def format_sexp_rec(sexp, buf):
```

```
    if type(sexp) == type('atom'):
        buf.fromstring(sexp)
    else:
        buf.fromstring('(')
        for i in range(len(sexp)):
            format_sexp_rec(sexp[i], buf)
            if i < len(sexp) - 1: buf.fromstring(' ')
        buf.fromstring(')')

def format_sexp(sexp):
    buf = array.array('c')
    format_sexp_rec(sexp, buf)
    return buf.tostring()

def dump_stack(sout, stack):
    for type, val in stack:
     if type == 'white':
         sout.write(val)
     else:
         sout.write(format_sexp(val))

def termsexp_proof(s, sout, termmap, varmap):
    stack = []
    while 1:
     tok = s.get_tok()
     if tok == None:
         raise SyntaxError('eof inside proof')
     elif tok == ')':
         dump_stack(sout, stack)
         sout.write(tok[-1])
         return
     if tok == '(':
         raise SyntaxError('need to deal with sexps in proofs')
     if tok[0] == 'id':
         id = tok[1]
         if termmap.has_key(id):
          nargs = termmap[id]
          args = []
          ix = len(stack)
          while len(args) < nargs:
              if ix == 0: raise SyntaxError('stack underflow')
              ix -= 1
              if stack[ix][0] == 'term':
               args.append(stack[ix][1])
          args.append(id)
          args.reverse()
          stack[ix:] = [('term', args)]
         elif varmap.has_key(id):
          stack.append(('term', id))
         else:
          dump_stack(sout, stack)
          stack = []
          sout.write(id)
     elif tok[0] == 'white':
```

```
          stack.append(tok)

def termsexp(sin, sout):
    s = Scanner(sin)
    termmap = {}
    varmap = {}
    while 1:
     tok = s.get_tok()
     if tok == None:
         break
     if tok[0] == 'id':
         cmd = tok[1]
         sout.write(cmd)
         if cmd == 'def':
          args = echo_sexp(s, sout)
          termmap[args[0][0]] = len(args[0]) - 1
         elif cmd == 'var':
          args = echo_sexp(s, sout)
          for v in args[1:]:
              varmap[v] = 1
         elif cmd == 'import':
          args = echo_sexp(s, sout)
          fn = args[1] + '.ghi'
          pref = args[3]
          if pref[0] == '"': pref = pref[1:-1]
          termsexp_import(fn, pref, termmap)
         elif cmd == 'thm':
          tok = echo_tok_nonwhite(s, sout)
          if tok != '(': raise SyntaxError('expected list after thm')
          echo_sexp(s, sout) # thm name
          echo_sexp(s, sout) # dv
          echo_sexp(s, sout) # hyps
          echo_sexp(s, sout) # concl
          tok = echo_tok_nonwhite(s, sout)
          if tok != '(': raise SyntaxError('expected list for proof')
          termsexp_proof(s, sout, termmap, varmap)
          tok = echo_tok_nonwhite(s, sout)
          if tok != ')': raise SyntaxError('expected nothing after proof')
         else:
          echo_sexp(s, sout)
      elif tok[0] == 'white':
         sout.write(tok[1])

if __name__ == '__main__':
    termsexp(sys.stdin, sys.stdout)
```