Answer Set 1

OUT: 20 SEPTEMBER, 1994

```
-----------ANSWER1.PAS-----------------------

[ENVIRONMENT]
module MyStringSoln1(input,output) ;

(* Solution for problem set 1,
   a module for strings
   (c) Burt Rosenberg, Univ. of Miami 1994
*)

const
  MYSTRINGLEN = 200 ;
  MYEOF = -1 ;

type
  MyString = record
             length : integer ;
             data : array [1..MYSTRINGLEN+1] of char ;
             end ;

procedure writeln_MyString( ms : myString ) ;
(*
   Writes the string ms, followed by a newline.
*)
var i : integer ;
begin
   i := 0 ;
   while ( i<ms.length ) do begin
      i := i + 1 ;
      write(ms.data[i]) ;
   end ;
   writeln ;
end ;

function readln_MyString( var ms : myString ) : integer ;
(*
   Reads the string into ms, up to a new line.
   Strings to long are truncated and error reported.
   Returns:
      the length of the string, 0 for an empty line
```

```
      MYEOF, and an empty string, if end of file encountered.
*)
begin
   ms.length := 0 ;
   if ( not eof ) then begin
      while ( not eoln ) do begin
         if ( ms.length < MYSTRINGLEN ) then
            ms.length := ms.length + 1
         else writeln('readln_MyString: string too long') ;
         read(ms.data[ms.length]) ;
      end ;
      readln ;
      readln_MyString := ms.length ;
   end else readln_MyString := MYEOF ;
end ;

function length_MyString( s: MyString ) : integer ;
begin
   length_MyString := s.length
end ;

procedure parse_MyString( var src: MyString; var oneWord: MyString) ;
(*  the first word is taken off of src and placed in oneWord.
    if there is no more words in src, oneWord is returned with
    the empty string. src is updated, removing oneWord, i.e.
    successive calls to parse_MyString will give successive
    words from src. *)

    function IsAChar( c: char ) : boolean ;
    begin
       IsAChar := (('A' <= c) AND (c <='Z'))
                  OR (('a' <= c) AND (c <= 'z'))
                  OR (('0' <= c) AND (c <= '9'))
                  OR ( '-' = c )
    end ;
var
   i, j : integer ;
begin
   (* phase 1, skip over non-characters at head of src *)
   (* put a sentinel in src. *)
   src.data[src.length+1] := 'X' ;
   i := 1 ;
   (* Loop Invariant: for all 0<j<i, src.data[j] is not a char. *)
   while ( NOT IsAChar(src.data[i]) ) do
```

```
      i := i + 1 ;

   (* phase 2, copy characters from src to oneWord *)
   (* put a sentinel in src. *)
   src.data[src.length+1] := ' ' ;
   j := 1 ;
   (* Loop Invariants:
         Let I be the index of the first letter to
            copy from src.data, or the sentinel.
         for all 0<k<j,
            1) oneWord.data[k] is valid
            2) it equals src.data[I+k-1]
   *)
   while ( IsAChar( src.data[i]) ) do begin
      oneWord.data[j] := src.data[i] ;
      j := j + 1 ;
      i := i + 1 ;
   end {while} ;
   oneWord.length := j - 1 ;

   (* phase 3, copy remainder of src forward in src *)
   j := 1 ;
   (* Loop Invariant:
         i points to next letter to copy, unless i too large
         j points to next place to copy to.
   *)
   while ( i <= src.length ) do begin
      src.data[j] := src.data[i] ;
      i := i + 1 ;
      j := j + 1 ;
   end {while} ;
   src.length := j - 1 ;

end ;

end.

----------------------DRIVER1.PAS--------------


[INHERIT('ANSWER1')]
program driver1( input, output ) ;

(*  driver program to test answer1.pas
```

```
   (c) Burton Rosenberg, U. of Miami 1994
*)

var
   a, b : MyString  ;
   i : integer ;

begin
   i := readln_MyString( a ) ;
   while ( i<>MYEOF ) do begin
      parse_MyString( a, b ) ;
      while ( length_MyString( b ) > 0 ) do begin
         writeln_MyString(b) ;
         parse_MyString( a, b ) ;
      end {while length ...} ;
      i := readln_MyString( a ) ;
   end { while not MYEOF } ;
end .

---------------------TEST1.RUN--------------

$ pascal answer1
$ pascal driver1
$ link driver1, answer1
$ run driver1
Romeo, Romeo!   wherefore
art    thou,    Romeo   ?
```

Sample Runs

```
$ spawn/input=test1.run
%DCL-S-SPAWNED, process BURT_1 spawned
%DCL-S-ATTACHED, terminal now attached to process BURT_1
Romeo
Romeo
wherefore
art
thou
Romeo
%DCL-S-RETURNED, control returned to process BURT
$
```