Recursion Examples

OUT: 4 OCTOBER, 1994

```
-------------------- example8 ----------------------

program example8(input,output) ;

(* Math 220/317 example pascal program
   Comparing For, While, Repeat and Recursion.
   (c) Burton Rosenberg 1994
   Univ. of Miami
*)

const N = 10 ;
type  MyArray = array [1..N] of char ;

procedure for_init( var a : MyArray ) ;
var i : integer ;
begin
   for i := 1 to N do a[i] := 'F'
end ;

procedure while_init( var a : MyArray ) ;
var i : integer ;
begin
   i := 1 ;
   (* L.I. a[j]=j for all 0<j<i *)
   while ( i<=N ) do begin
      a[i] := 'W' ;
      i := i + 1
   end
   (* L.I. + Termination = GOAL, array initialized *)
end ;

procedure repeat_init( var a : MyArray ) ;
var i : integer ;
begin
   if N>0 then begin
     i := 1 ;
     (* L.I. a[j]=j for all 0<j<i *)
     repeat
        a[i] := 'R' ;
        i := i + 1 ;
```

```
     until i>N
   (* L.I. + Termination = GOAL, array initialized *)
   end
end ;

procedure recursive_init( var a : MyArray; i : integer ) ;
begin
   (* Recursion Invariant: initialize array a, locations
      a[1] to a[i] *)
   if i>0 then begin
      a[i] := 'X' ;
      recursive_init( a, i-1 )
   end
end ;

var
   array_to_init : MyArray ;
begin
   for_init(array_to_init) ;
   while_init(array_to_init) ;
   repeat_init(array_to_init) ;
   recursive_init(array_to_init, N ) ;
end.

-------------------- example9 ----------------------

[INHERIT('mystring2','mylist1')]

program example9(input,output) ;

(* example of listing a list by recursion. *)

procedure recursive_print_mylist( ml : MyList ) ;
begin
(* Recursion Invariant: ml is a mylist to print *)
   if (ml<>NIL) then begin
      (* write this element *)
      write(ml^.count:3,' ') ;
      writeln_MyString(ml^.theMs) ;
      (* then recurse *)
      recursive_print_mylist( ml^.next )
   end
(* Postcondition, ml is printed *)
end ;
```

```
procedure recursive_RevPrint_mylist( ml : MyList ) ;
begin
(* Recursion Invariant: ml is a mylist to print *)
   if (ml<>NIL) then begin
      (* recurse on tail *)
      recursive_RevPrint_mylist( ml^.next ) ;
      (* then write this element *)
      write(ml^.count:3,' ') ;
      writeln_MyString(ml^.theMs)
   end
(* Postcondition, ml is printed *)
end ;

var
  thelist : MyList ;

begin
   create_MyList( thelist ) ;
   input_routine( thelist ) ;
   recursive_print_MyList( thelist ) ;
   recursive_RevPrint_MyList( thelist ) ;

end.

-------------------- example10 ---------------------

[INHERIT('mystring2')]
program example10(input,output) ;

(* comparing two strings using Recursion.
   (c) Burton Rosenberg 1994
   Univ. of Miami
*)

(* The following are declared in mystring2.pas
const
    MYSTRINGLEN = 200 ;
    MYEOF = -1 ;
type
    MyString = record
               length : integer ;
               data : array [1..MYSTRINGLEN+1] of char ;
               end ;
```

```
*)

function Compare_MyString( a: MyString; b: MyString ) : integer ;
(*  compares strings a and b,
    returns 1 if a<b,
            0 if a=b
           -1 if a>b            *)

    function clip_value( i: integer ): integer ;
    begin
       if (i>1) then i := 1
       else if (i<-1) then i := -1 ;
       clip_value := i
    end ;

    function Comp_MyString_Aux( i : integer ) : integer ;
    (* Precondition a.length<=b.length *)
    (* R.I. a.data[j]=b.data[j] for all 0<j<i *)
    begin
       if (i>a.length) then
          Comp_MyString_Aux := clip_value( b.length - i + 1 )
       else if a.data[i]=b.data[i] then
          Comp_MyString_Aux := Comp_MyString_Aux(i+1)
       else Comp_MyString_Aux := clip_value( ord(b.data[i])-ord(a.data[i]) )
    end ;

begin
   if a.length>b.length then Compare_MyString := - Compare_MyString(b,a)
   else Compare_MyString := Comp_MyString_Aux( 1 )
end ;

var a, b : MyString ;
    i : integer ;
begin
    write('Enter string A: ') ;
    i := Readln_MyString(a) ;
    write('Enter string B: ') ;
    i := Readln_Mystring(b) ;
    i := Compare_MyString(a,b) ;
    write('Compare_MyString(A,B) returns ',i:2) ;
end.
```