Burt Rosenberg
program divisors;
{
  Burton Rosenberg
  Homework 1 from Math 220/317, Fall 1993.
  University of Miami

  Problems 5-24, 5-25 pages 182, 183 from
  Oh! Pascal! Turbo Pascal 6.0 by Doug Cooper.
}

const
  PrWidth = 5;

{============= getDivisors ===============}

procedure getDivisors( n : integer ; var num, sum : integer ;
  printing : boolean ) ;
{ get all divisors of n,
  input:
    n, the number to get divisors of
    printing, true if procedure should print divisors
  output:
    sum, the sum of divisors of n
    num, the number of divisors of n
}

var i : integer;
begin
  {1 divides every number, and we care only about proper
divisors, so skip 1 in loop but count in sum and num here.}
  sum := 1; {the sum of divisors}
  num := 1; {the number of divisors}
  if (printing) then write( sum:PrWidth ); {sum is just a convenient 1}
i := 2;

  {loop through all i strictly less than square root of n}
  while (i*i < n) do begin
    if ( n mod i = 0 ) then begin { i divides n }
      sum := sum + i + (n div i);
      num := num + 2;
    end;
  end;

begin

end.
if (printing) then 
    write( i:PrWidth, (n div i):PrWidth ) ;
end ;
i := i + 1 ;
end ;

{take care if n is a perfect square}
if (i*i = n) then begin
    num := num + 1 ;
    sum := sum + i ;
    if (printing) then
        write( i:PrWidth ) ;
end ;
end ;

{=============== various routines ===============}

decision tableOfDiv( upto : integer ) ;
{ prints a table of the divisors of num for
  num from 1 to upto. }
var
  i : integer ;
x, y : integer ;
begin
  writeln('Num   Divisors') ;
  writeln('---------------------------------------------') ;
  for i := 1 to upto do begin
      write( i:PrWidth ) ;
      getDivisors( i, x, y, true ) ;
      writeln ;
  end ;
end ;

decision mostDivisible( upto : integer ) ;
{ finds the integer from 1 to upto which has the
  largest number of divisors. }
var
  i, j, k : integer ;
  mostDiv : integer ;
  numDiv : integer ;
begin
  mostDiv := 1 ;
numDiv := 1;
i := 1;
{ LOOP INVARIANT:
    the most divisible number i or less is
    mostDiv, and it has numDiv divisors.}
{ ASSERT: loop Invariant }
for i:= 2 to upto do begin
    getDivisors( i, j, k, false );
    if (j>numDiv) then begin { the current i is more divisible }
        mostDiv := i;
        numDiv := j;
    end;
{ ASSERT: loop invariant }
end;
{TERMINATION assured, it was a for loop!}
{GOAL = TERMINATION + INVARIANT, we have the most divisible
    number from1 to upto. }
writeln('The most divisible number from 1 to ', upto:PrWidth,
    ' is ',mostDiv:PrWidth,', it has ',numDiv:PrWidth,' divisors.');
end;

procedure findPerfect( upto : integer ) ;
{ Looks for number from 2 to upto whose divisors
    sum to the number itself. }
var
    i, j, k : integer ;
begin
    for i := 2 to upto do begin {1 is not considered perfect.}
        getDivisors( i, j, k, false ); {get the divisors}
        if (k=i) then begin {if number is perfect}
            write(i:PrWidth,' is perfect. It has divisors: ');
            getDivisors( i, j, k, true ); {print the divisors}
            writeln ;
            end;
    end;
end;

procedure findOddAbundant( upto : integer ) ;
{ looks for odd number from 3 to upto whose divisors
sum to greater than the number itself.
}
var
i, j, k : integer ;
begin

{use a while loop, since we just want odd numbers }
i := 3 ;
while ( i <= upto ) do begin

    getDivisors( i, j, k, false ) ; {get the divisors}
    if (k>i) then begin {if number is abundant}
        writeln(i:PrWidth,' is abundant. It has divisors: ') ;
        getDivisors( i, j, k, true ) ; {print the divisors}
    end ;
    i := i + 2 ;
end ;

end ;

{================MAIN LINE=================}

var
num : integer ;
i,j : integer ;
begin

{ For problem 5-24 (a):
    writeln('What number to find divisors') ;
    readln(num) ;
    if (num>0) then
        getDivisors( num, i, j, true )
    else writeln('Number must be strictly positive.') ;
    writeln ;
}

{ For problem 5-24 (b):
    write('Size of table: ') ;
    readln(num) ;
    tableOfDiv( num ) ;
}

{ For problem 5-24 (c):
    writeln('Number to search until') ;
readln(num);
mostDivisible(num);
}
{ For problem 5-25 (a):
write('Find perfect numbers up to what size: ');
readln(num);
findPerfect(num);
}
{ For problem 5-25 (b):
}
write('Find odd abundant numbers up to what size: ');
readln(num);
findOddAbundant(num);
end.

Sample Runs

impala> a.out
Size of table: 30
Num Divisors
---------------------------------------------
 1 1
 2 1
 3 1
 4 1 2
 5 1
 6 1 2 3
 7 1
 8 1 2 4
 9 1 3
10 1 2 5
11 1
12 1 2 6 3 4
13 1
14 1 2 7
15 1 3 5
16 1 2 8 4
17 1
18 1 2 9 3 6
19 1
20 1 2 10 4 5
21 1 3 7
Math 220/317: Programming II/Data Structures

```plaintext
22 1 2 11
23 1
24 1 2 12 3 8 4 6
25 1 5
26 1 2 13
27 1 3 9
28 1 2 14 4 7
29 1
30 1 2 15 3 10 5 6

impala> a.out
Find perfect numbers up to what size: 1000
   6 is perfect. It has divisors: 1 2 3
 28 is perfect. It has divisors: 1 2 14 4 7
496 is perfect. It has divisors: 1 2 248 4 124 8 62 16 31

impala> a.out
Find odd abundant numbers up to what size: 1000
  945 is abundant. It has divisors: 1 3 315 5 189 7 135 9 105 15 63
  21 45 27 35
```