Solutions to Assignment 3

Question 1

The following solution uses the high-level design suggested in the assignment. The abbreviation X"nn" allows bit_vector constants to be specified in hexadecimal notation.

```
-- ELEC 464 Assignment 3
-- Ed Casas, September 25 1996

entity snum is
  port ( clk : in bit ;
a,b,c,d,e,f,g : out bit ) ;

-- pragma dc_script_begin
-- set_port_is_pad "*"
-- set_attribute clk pad_location -type string "P18"
-- set_pad_type -no_clock clk
-- set_attribute a pad_location -type string "P49"
-- set_attribute b pad_location -type string "P48"
-- set_attribute c pad_location -type string "P47"
-- set_attribute d pad_location -type string "P46"
-- set_attribute e pad_location -type string "P45"
-- set_attribute f pad_location -type string "P50"
-- set_attribute g pad_location -type string "P51"
-- pragma dc_script_end

end snum ;


architecture rtl of snum is
   signal i, nexti : bit_vector (3 downto 0) ;
   signal digit : bit_vector (3 downto 0) ;
   signal seg : bit_vector (6 downto 0) ;
begin

   -- counter combinational logic
   process(i)
   begin
      case i is
         when X"0" => nexti <= X"1" ;
         when X"1" => nexti <= X"2" ;
         when X"2" => nexti <= X"3" ;
         when X"3" => nexti <= X"4" ;
         when X"4" => nexti <= X"5" ;
         when X"5" => nexti <= X"6" ;
         when X"6" => nexti <= X"7" ;
         when X"7" => nexti <= X"8" ;
         when X"8" => nexti <= X"0" ;
         when others => nexti <= X"0" ;
      end case ;
   end process ;


   -- counter sequential logic
   process(clk,nexti)
   begin
      if clk'event and clk='0' then
         i <= nexti ;
      end if ;
   end process ;


   -- convert digit index (i) into the i'th digit
   -- of the student number
   process(i)
   begin
      case i is
         when X"0" => digit <= X"F" ;
         when X"1" => digit <= X"8" ;
         when X"2" => digit <= X"7" ;
         when X"3" => digit <= X"6" ;
         when X"4" => digit <= X"5" ;
         when X"5" => digit <= X"4" ;
         when X"6" => digit <= X"3" ;
         when X"7" => digit <= X"2" ;
         when X"8" => digit <= X"1" ;
         when others => digit <= X"F" ;
      end case ;
   end process ;


   -- convert a digit into on/off values for
   -- a 7-segment display
   process(digit)
   begin
      case digit is
         when X"0" => seg <= "1111110" ;
         when X"1" => seg <= "0110000" ;
         when X"2" => seg <= "1101101" ;
         when X"3" => seg <= "1111001" ;
         when X"4" => seg <= "0110011" ;
         when X"5" => seg <= "1011011" ;
         when X"6" => seg <= "1011111" ;
         when X"7" => seg <= "1110000" ;
         when X"8" => seg <= "1111111" ;
         when X"9" => seg <= "1110011" ;
         -- blank display for invalid values
         when others => seg <= "0000000" ;
      end case ;
   end process ;


   -- copy 'seg' vector to individual output ports
   -- need to negate because LED outputs are active-low
   process(seg)
   begin
      a <= not seg(6) ;
      b <= not seg(5) ;
      c <= not seg(4) ;
      d <= not seg(3) ;
      e <= not seg(2) ;
      f <= not seg(1) ;
      g <= not seg(0) ;
   end process ;

end rtl ;
```