Solutions to Mid-Term Exam

Question 1 (a)

```
-- Decade counter with enable input and
-- terminal count output.
-- Ed Casas, February 25, 1999

library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;

entity decade is port (
   en, clk : in std_logic ; -- enable and clock
   q : out std_logic_vector (3 downto 0) ; -- count
   tc : out std_logic ) ; -- terminal count
end decade ;

architecture rtl of decade is
   signal count, next_count, count_plus_1 :
               unsigned (3 downto 0) ;
begin
   -- next value in count
   count_plus_1 <=
        0 when count = 9 else
        count + 1 ;

   -- next count value
   next_count <=
        count_plus_1 when en = '1' else
        count ;

   -- register the count
   process(cp)
   begin
        if cp'event and cp = '1' then
           count <= next_count ;
        end if ;
   end process ;

   -- connect to output
   q <= std_logic_vector(count) ;

   -- terminal count
   tc <=
        '1' when count = 9 else
        '0' ;

end rtl ;
```

Question 1 (b)

```
-- Two-digit decimal counter.
-- Ed Casas, February 25, 1999

library ieee ;
use ieee.std_logic_1164.all ;
use work.counters.all ;

entity twodigits is port (
        clk : in std_logic ;
        a, b : out std_logic_vector (3 downto 0) ) ;
end decade ;

architecture rtl of twodigits is
   signal a_en, b_en, b_tc : std_logic ;
begin
   a_en <= '1' ;
   c1: decade port map ( a_en, clk, a, b_en ) ;
   c2: decade port map ( b_en, clk, b, b_tc ) ;
end rtl ;
```

Question 2

```
;
; ELEC 379 1998/99 Mid-Term Exam
; Ed Casas, February 25, 1999
;
; convert string to lower case
;
toupper:
        push    ax      ; save registers
        push    bx
loop:   mov     al,[bx] ; get a character
        cmp     al,0    ; stop if it's zero
        jz      done
        cmp     al,'a'  ; convert if between
        jb      nocon   ;    'a' and 'z'
        cmp     al,'z'
        ja      nocon
        sub     al,20H  ; convert to lower case
        mov     [bx],al
nocon:  inc     bx      ; repeat for next character
        jmp     loop
done:   pop     bx      ; restore registers
        pop     ax
        ret
```