ESERCIZI NEL LINGUAGGIO ASSEMBLATIVO DEL MICROCONTROLLORE PIC 16F84

1°) Calcolare in binario e convertire in decimale il contenuto del registro W dopo l'esecuzione delle seguenti istruzioni:

```
MOVLW     150  ;W = (10010110)2 and
ANDLW     170  ;    (10101010)2 =
               ;    (10000010)2 = 2 + 128 = 130
```

2°) Calcolare in binario e convertire in decimale il contenuto del registro W dopo l'esecuzione delle seguenti istruzioni:

```
MOVLW     45   ;W = (00101101)2 ior
IORLW     180  ;    (10110100)2 =
               ;    (10111101)2 = 1 + 4 + 8 + 16 +
               ;                  32 + 128 = 189
```

3°) Calcolare in binario e convertire in decimale il contenuto del registro W dopo l'esecuzione delle seguenti istruzioni:

```
MOVLW     60   ;W = (00111100)2 xor
XORLW     165  ;    (10100101)2 =
               ;    (10011001)2 = 1 + 8 + 16 + 128 =
               ;                = 153
```

4°) Calcolare in binario e convertire in decimale il contenuto del registro W ed il valore dei *flag* C, DC e Z dopo l'esecuzione delle seguenti istruzioni:

```
               ;        11011001
MOVLW     205  ;    W = (11001101)2 +
ADDLW     217  ;      (11011001)2 =
               ;        (10100110)2 = 2 + 4 + 32 +
               ;                    + 128 = 166
               ;    C = 1, DC = 1, Z = 0
```

5°) Calcolare in binario e convertire in decimale il contenuto del registro W ed il valore dei *flag* C, DC e Z dopo l'esecuzione delle seguenti istruzioni:

```
               ;        11001111
MOVLW     237  ;    W = (10111000)2 -
SUBLW     184  ;      (11101101)2 =
               ;        (11001011)2 = 1 + 2 + 8 + 64
               ;                    + 128 = 203
               ;    C = 0, DC = 0, Z = 0
```

6°) Trasformare il contenuto del registro d'indirizzo $(20)_{16}$ dal codice ASCII di una lettera in forma maiuscola al codice ASCII della stessa lettera in forma minuscola.

```
MOVLW     20H       ;W = 20h
ADDWF     20H,F     ;[20h] = [20h] + 20h
```

7°) Copiare nel registro di lavoro W la rappresentazione binaria della cifra decimale il cui codice ASCII è contenuto nel registro d'indirizzo $(20)_{16}$.

```
MOVF      20H,W     ;W = [20H]
ANDLW     00001111B ;W = W AND 00001111B (bit a bit)
```

8°) Scambiare tra loro i contenuti dei registri d'indirizzo $(20)_{16}$ e $(30)_{16}$.

```
MOVF      20H,W     ;W = [20H]
MOVWF     40H       ;[40H] = W
MOVF      30H,W     ;W = [30H]
MOVWF     20H       ;[20H] = W
MOVF      40H,W     ;W = [40H]
MOWF      30H       ;[30H] = W
```

9°) Invertire il livello del segnale presente su ciascuna delle quattro linee d'uscita RB4, RB5, RB6 ed RB7, senza modificare il segnale sulle altre linee della porta B.

```
MOVLW     11110000b ;W = 11110000b
XORWF     PORTB,F   ;PORTB = PORTB XOR W
```

10°) Sommare al contenuto del registro d'indirizzo $(20)_{16}$ quello del registro d'indirizzo $(30)_{16}$ solo se il risultato è minore di 256.

```
MOVF      20H,W     ;W = [20₁₆].
ADDWF     30H,W     ;W = ([20₁₆] + [30₁₆]) mod 256.
BTFSS     STATUS,C  ;Se il riporto finale è zero,
MOVWF     20H       ;allora [20₁₆] = W.
```

11°) Sottrarre al contenuto del registro d'indirizzo $(20)_{16}$ quello del registro d'indirizzo $(30)_{16}$ solo se il risultato è maggiore o uguale a zero.

```
MOVF      30H,W     ;W = [30₁₆].
SUBWF     20H,W     ;W = ([20₁₆] - [30₁₆]) mod 256.
BTFSC     STATUS,C  ;Se il prestito finale è zero,
MOVWF     20H       ;allora [20₁₆] = W.
```

12°) Traslare di una posizione verso sinistra tutti i bit del codice binario contenuto nel registro d'indirizzo $(20)_{16}$, copiando il valore del bit più significativo nella posizione di ordine zero.

```
MOVF      20H,W     ;W = [20H]
ADDWF     20H,F     ;[20H] = [20H] * 2
MOVF      STATUS,W  ;W = STATUS
ANDLW     00000001B ;W = Carry
ADDWF     20H,F     ;[20H] = [20H] + Carry
```

13°) Incrementare di uno il numero binario di 16 cifre la cui parte bassa è contenuta nel registro d'indirizzo $(20)_{16}$ e la cui parte alta è contenuta nel registro d'indirizzo $(21)_{16}$.

```
MOVLW     1          ;W = 1
ADDWF     20H,F      ;[20h] = [20h] + W = [20h] + 1
MOVF      STATUS,W   ;W = STATUS
ANDLW     00000001B  ;W = STATUS and (00000001)2
ADDWF     21H,F      ;[21h] = [21h] + Carry
```

14°) Moltiplicare per due il numero binario di 16 cifre la cui parte bassa è contenuta nel registro d'indirizzo $(20)_{16}$ e la cui parte alta è contenuta nel registro d'indirizzo $(21)_{16}$.

```
MOVF      20H,W      ;W = [20h]
ADDWF     20H,F      ;[20h] = [20h] + W = [20h] * 2
MOVF      STATUS,W   ;W = STATUS
ANDLW     00000001B  ;W = STATUS and (00000001)2
ADDWF     21H,W      ;W = [21h] + Carry
ADDWF     21H,F      ;[21h] = [21h] + W =
                     ;      = [21h] * 2 + Carry
```

15°) Configurare le linee RA0, RA1, RA2 ed RA3 in ingresso; la linea RA4 in uscita ed a livello alto.

```
BSF       STATUS,RP0     ;Seleziona TRISA.
MOVLW     00001111B      ;Imposta RA0..3 in ingresso
MOVWF     TRISA          ;Imposta RA4 in uscita.
BCF       STATUS,RP0     ;Seleziona PORTA.
BSF       PORTA,4        ;Poni RA4 a livello alto.
```

16°) Simulare una porta logica NOT con ingresso la linea RA0 ed uscita la linea RA4 (si supponga che RA0 sia già impostata in ingresso ed RA4 in uscita).

```
CICLO:    BTFSS     PORTA,0   ;Se RA0 è a livello basso
          GOTO      RA0L      ;allora vai a RA0L.
RA0H:     BCF       PORTA,4   ;Poni RA4 a livello basso.
          GOTO      CICLO     ;Vai a CICLO.
RA0L:     BSF       PORTA,4   ;Poni RA4 a livello alto.
          GOTO      CICLO     ;Vai a CICLO.
```

17°) Simulare una porta AND con ingressi le linee RA0 ed RA1 ed uscita la linea RA4 (si supponga che RA0 ed RA1 siano già impostate in ingresso ed RA4 in uscita).

```
CICLO:    BTFSC     PORTA,0   ;Se RA4 è a livello alto
          GOTO      RA0H      ;allora vai a RA0H.
RA0L:     BCF       PORTA,4   ;Poni RA4 a livello basso.
          GOTO      CICLO     ;Vai a CICLO.
RA0H:     BTFSS     PORTA,1   ;Se RA4 è a livello basso
          GOTO      RA0L      ;allora vai a RA0L.
          BSF       PORTA,4   ;Poni RA4 a livello alto.
          GOTO      CICLO     ;Vai a CICLO.
```

18°) Simulare una porta OR con ingressi le linee RA0 ed RA1 ed uscita la linea RA4 (si supponga che RA0 ed RA1 siano già impostate in ingresso ed RA4 in uscita).

```
CICLO:    BTFSS     PORTA,0   ;Se RA4 è a livello basso
          GOTO      RA0L      ;allora vai a RA0L.
RA0H:     BSF       PORTA,4   ;Poni RA4 a livello alto.
          GOTO      CICLO     ;Vai a CICLO.
RA0L:     BTFSC     PORTA,1   ;Se RA4 è a livello alto
          GOTO      RA0H      ;allora vai a RA0H.
          BCF       PORTA,4   ;Poni RA4 a livello basso.
          GOTO      CICLO     ;Vai a CICLO.
```

19°) Simulare una porta EXOR con ingressi le linee RA0 ed RA1 ed uscita la linea RA4 (si supponga che RA0 ed RA1 siano già impostate in ingresso ed RA4 in uscita).

```
CICLO:    BTFSS     PORTA,0   ;Se RA0 è a livello basso
          GOTO      RA0L      ;allora vai a RA0L.
RA0H:     BTFSS     PORTA,1   ;Se RA1 è a livello basso
          GOTO      RA1L      ;allora vai a RA1L.
RA1H:     BCF       PORTA,4   ;Poni RA4 a livello basso.
          GOTO      CICLO     ;Vai a CICLO.
RA0L:     BTFSS     PORTA,1   ;Se RA0 è a livello basso
          GOTO      RA1H      ;allora vai a RA1H.
RA1L:     BSF       PORTA,4   ;Poni RA4 a livello alto.
          GOTO      CICLO     ;Vai a CICLO.
```

20°) Simulare una porta NAND con ingressi le linee RA0 ed RA1 ed uscita la linea RA4 (si supponga che RA0 ed RA1 siano già impostate in ingresso ed RA4 in uscita).

```
CICLO:    BTFSS     PORTA,0   ;Se RA0 è a livello basso
          GOTO      HIGH      ;allora vai a HIGH.
          BTFSS     PORTA,1   ;Se RA1 è a livello basso
          GOTO      HIGH      ;allora vai a HIGH.
LOW:      BCF       PORTA,4   ;Poni RA4 a livello basso.
          GOTO      CICLO     ;Vai a CICLO.
HIGH:     BSF       PORTA,4   ;Poni RA4 a livello alto.
          GOTO      CICLO     ;Vai a CICLO.
```

21°) Simulare una porta NOR con ingressi le linee RA0 ed RA1 ed uscita la linea RA4 (si supponga che RA0 ed RA1 siano già impostate in ingresso ed RA4 in uscita).

```
CICLO:    BTFSC     PORTA,0   ;Se RA0 è a livello alto
          GOTO      LOW       ;allora vai a LOW.
          BTFSC     PORTA,1   ;Se RA1 è a livello alto
          GOTO      LOW       ;allora vai a HIGH.
HIGH:     BSF       PORTA,4   ;Poni RA4 a livello alto.
          GOTO      CICLO     ;Vai a CICLO.
LOW:      BCF       PORTA,4   ;Poni RA4 a livello basso.
          GOTO      CICLO     ;Vai a CICLO.
```

22°) Simulare il funzionamento di un “latch RS” con ingressi RA0 = SET, RA1 = RESET ed uscite RA2 = Q, RA3 = ~Q.

```
            BSF       STATUS,RP0     ;Seleziona TRISA.
            MOVLW     00010011B      ;Imposta in input RA0 e RA1
            MOVWF     TRISA          ;in output RA2 ed RA3.
            BCF       STATUS,RP0     ;Seleziona PORTA.
S0:         BCF       PORTA,2        ;Poni Q a livello basso.
            BSF       PORTA,3        ;Poni ~Q a livello alto.
            BTFSC     PORTA,0        ;Se Set è alto allora
            GOTO      S1             ;vai a S1, altrimenti
            GOTO      S0             ;resta in S0.
S1:         BSF       PORTA,2        ;Poni Q a livello alto.
            BCF       PORTA,3        ;Poni ~Q a livello basso.
            BTFSC     PORTA,1        ;Se RESET è alto allora
            GOTO      S0             ;vai a S0, altrimenti
            GOTO      S1             ;resta in S1.
```

23°) Simulare il funzionamento di un “latch D” con ingressi RA0 = DATA, RA1 = ENABLE ed uscite RA2 = Q, RA3 = ~Q.

```
            BSF       STATUS,RP0     ;Seleziona TRISA.
            MOVLW     00010011B      ;Imposta in input RA0 e RA1
            MOVWF     TRISA          ;in output RA2 ed RA3.
            BCF       STATUS,RP0     ;Seleziona PORTA.
S0:         BCF       PORTA,2        ;Poni Q a livello basso.
            BSF       PORTA,3        ;Poni ~Q a livello alto.
            BTFSS     PORTA,1        ;Se ENABLE è basso allora
            GOTO      S0             ;resta in S0, altrimenti
            BTFSC     PORTA,0        ;se DATA è alto allora
            GOTO      S1             ;vai a S1, altrimenti
            GOTO      S0             ;resta in S0.
S1:         BSF       PORTA,2        ;Poni Q a livello alto.
            BCF       PORTA,3        ;Poni ~Q a livello basso.
            BTFSS     PORTA,1        ;Se ENABLE è basso allora
            GOTO      S1             ;resta in S1, altrimenti
            BTFSC     PORTA,0        ;se DATA è alto allora
            GOTO      S1             ;resta in S1, altrimenti
            GOTO      S0             ;vai a S0.
```

24°) Scrivere una procedura di nome AZZERA che azzeri tutti i registri generali del microcontrollore PIC16F84.

```
AZZERA:     MOVLW     0CH            ;W = (0C)16
            MOVWF     FSR            ;FSR = W = (0C)16
CICLO:      MOVLW     0              ;W = 0
            MOVWF     INDF           ;[FSR] = W = 0
            MOVLW     1              ;W = 1
            ADDWF     FSR            ;FSR = FSR + 1
            MOVLW     50H            ;W = (50)16
            SUBWF     FSR,W          ;W = FSR - (50)16
            BTFSS     STATUS,Z       ;Se FSR <> (50)16 allora
            GOTO      CICLO          ;ripeti il ciclo altrimenti
            RETURN                   ;termina la procedura.
```

25°) Scrivere una procedura di nome INC_ASCII che incrementi di uno (modulo 10) la cifra decimale il cui codice ASCII è memorizzato nel registro il cui indirizzo è contenuto nel registro FSR, restituendo l'eventuale riporto nel registro di lavoro W.

```
INC_ASCII:INCF      INDF          ;[FSR] = [FSR] + 1.
          MOVLW     3AH           ;W = ASCII(9) + 1.
          SUBWF     INDF,W        ;W = [FSR] - (ASCII(9)+1).
          BTFSS     STATUS,Z      ;Se [FSR] ≠ (ASCII(9)+1)
          RETLW     0             ;termina la procedura con
                                  ;W = 0 (nessun riporto).
          MOVLW     30H           ;altrimenti:
          MOVWF     INDF          ;[FSR] = ASCII(0).
          RETLW     1             ;Termina la procedura con
                                  ;W = 1 (riporto di uno).
```

26°) Scrivere una procedura di nome INC_NUM che incrementi di uno il numero decimale di “n” cifre, con “n” rappresentato in binario nel registro W, il cui codice ASCII è memorizzato a partire dal registro il cui indirizzo è contenuto nel registro FSR, restituendo l'eventuale riporto nel registro di lavoro W.

```
CONTA     EQU       20H           ;Indirizzo di CONTA = 20₁₆.
INC_NUM:  MOVWF     CONTA         ;CONTA = W = n.
CICLO:    CALL      INC_ASCII     ;[FSR] = [FSR]+1 modulo 10.
          SUBLW     1             ;Z = W = riporto.
          BTFSS     STATUS,Z      ;Se “nessun riporto” allora
          RETLW     0             ;termina la procedura con
                                  ;W = 0 (nessun riporto).
          INC       FSR           ;FSR = FSR + 1.
          DECFSZ    CONTA         ;CONTA = CONTA - 1.
          GOTO      CICLO         ;Se CONTA ≠ 0 vai a CICLO.
          RETLW     1             ;Termina la procedura con
                                  ;W = 1 (riporto di uno).
```

27°) Scrivere una procedura di nome INC_BCD che incrementi di uno (modulo 100) il numero decimale di due cifre il cui codice BCD è memorizzato nel registro il cui indirizzo è contenuto nel registro FSR, restituendo l'eventuale riporto nel registro W.

```
INC_BCD:  INCF      INDF          ;[FSR] = [FSR] + 1.
          MOVLW     0FH           ;W = (00001111)₂.
          ANDWF     INDF,W        ;W = [FSR]₀,,₃.
          SUBLW     0AH           ;W = 10 - W.
          BTFSS     STATUS,Z      ;Se [FSR]₀,,₃ ≠ 10 allora
          RETLW     0             ;termina la procedura con
                                  ;W = 0 (nessun riporto).
          MOVLW     6             ;Se [FSR]₀,,₃ = 10 allora
          ADDWF     INDF          ;[FSR] = [FSR] + 6.
          MOVLW     0A0H          ;W = 10 x 2⁴.
          SUBWF     INDF,W        ;W = [FSR] - 10 x 2⁴.
          BTFSS     STATUS,Z      ;Se [FSR] ≠ 100 allora
          RETLW     0             ;termina la procedura con
                                  ;W = 0 (nessun riporto).
          CLRF      INDF          ;[FSR] = 0.
          RETLW     1             ;Termina la procedura con
                                  ;W = 1 (riporto di uno).
```

28°) Scrivere una procedura di nome INC_NUM che incrementi di uno il numero decimale di “2 x n” cifre, con “n” rappresentato in binario nel registro W, il cui codice BCD è memorizzato a partire dal registro il cui indirizzo è contenuto nel registro FSR, restituendo l'eventuale riporto nel registro di lavoro W.

```
CONTA     EQU      20H           ;Indirizzo di CONTA = 20₁₆.
INC_NUM:  MOVWF    CONTA         ;CONTA = W = n.
CICLO:    CALL     INC_BCD       ;[FSR] = [FSR]+1 modulo 100
          SUBLW    1             ;Z = W = riporto.
          BTFSS    STATUS,Z      ;Se "nessun riporto" allora
          RETLW    0             ;termina la procedura con
                                 ;W = 0 (nessun riporto).
          INC      FSR           ;FSR = FSR + 1.
          DECFSZ   CONTA         ;CONTA = CONTA – 1.
          GOTO     CICLO         ;Se CONTA ≠ 0 vai a CICLO.
          RETLW    1             ;Termina la procedura con
                                 ;W = 1 (riporto di uno).
```

29°) Scrivere una procedura di nome DEC_ASCII che decrementi di uno (modulo 10) la cifra decimale il cui codice ASCII è memorizzato nel registro il cui indirizzo è contenuto nel registro FSR, restituendo l'eventuale prestito nel registro di lavoro W.

30°) Scrivere una procedura di nome DEC_NUM che decrementi di uno il numero decimale di “n” cifre, con “n” rappresentato in binario nel registro W, il cui codice ASCII è memorizzato a partire dal registro il cui indirizzo è contenuto nel registro FSR, restituendo l'eventuale prestito nel registro di lavoro W.

31°) Scrivere una procedura di nome DEC_BCD che incrementi di uno (modulo 100) il numero decimale di due cifre il cui codice BCD è memorizzato nel registro il cui indirizzo è contenuto nel registro FSR, restituendo l'eventuale prestito nel registro di lavoro W.

32°) Scrivere una procedura di nome DEC_NUM che decrementi di uno il numero decimale di “2 x n” cifre, con “n” rappresentato in binario nel registro W, il cui codice BCD è memorizzato a partire dal registro il cui indirizzo è contenuto nel registro FSR, restituendo l'eventuale prestito nel registro di lavoro W.

33°) Supponendo che la frequenza del segnale CLK_{IN} del microcontrollore sia pari a 20 Mhz, scrivere una procedura di nome RITARDO la cui esecuzione richieda un periodo di tempo della durata di circa 10 millisecondi.

Soluzione senza l'uso del registro TIMER0:

```
;Poiché alla frequenza di 20 Mhz il microcontrollore PIC
;esegue 5 milioni di istruzioni al secondo, in un periodo di
;10 millisecondi eseguirà 5x10^6x10x10^-3 = 50.000 istruzioni.
;Affinché la seguente procedura richieda l'esecuzione di tale
;numero di istruzioni è necessario inizializzare il registro
;CONTA al valore 50.000 /(3 x 256 + 2) ≈ 65:
```

```
RITARDO:  MOVLW     65
          MOVWF     CONTA      ;CONTA = 65.
          CLRW                 ;W = 0.
CICLO:    ADDLW     1          ;W = (W + 1) mod 256.
          BTFSZ     STATUS,Z   ;Se W ≠ 0 allora
          GOTO      CICLO      ;ripeti il ciclo.
          DECFSZ    CONTA      ;CONTA = CONTA - 1.
          GOTO      CICLO      ;Se CONTA ≠ 0 torna a CICLO
          RETURN               ;altrimenti termina la procedura
```

Soluzione con l'uso del registro TIMER0:

```
RITARDO:  BSF       STATUS,5   ;Seleziona il banco 1.
          MOVLW     00000111B  ;Inizializza il registro OPTION
          MOVWF     OPTION     ;in modo che PRESCALER = 7.
          BCF       STATUS,5   ;Seleziona il banco 0.
          MOVLW     61         ;Inizializza TIMER0:
          MOVWF     TIMER0     ;TIMER0 = 256 - 195 = 61
          BCF       INTCON,2   ;T0IF = 0.
CICLO:    BTFSS     INTCON,2   ;Ciclo d'attesa
          GOTO      CICLO      ;dell'evento T0IF = 1.
          RETURN               ;Fine della procedura.

;L'esecuzione della precedente procedura richiede un periodo
;di tempo pari a T_CLK x 4 x 256 x 195 ≈ 10 ms.
```

34°) Scrivere il programma di gestione di un microcontrollore PIC 16F84 che opera alla frequenza di 20 Mhz e che produce, sulla linea d'uscita RA4, un'onda quadra di frequenza pari a 500 Hz.

```
INIZIO:   BSF       STATUS,RP0;Seleziona il banco 1.
          MOVLW     00001111B
          MOVWF     TRISA      ;RA4 = OUTPUT.
          MOVLW     00000100B  ;Inizializza il registro OPTION
          MOVWF     OPTION     ;in modo che PRESCALER = 4.
          BCF       STATUS,RP0;Seleziona il banco 0.
CICLO:    MOVLW     100        ;Inizializza TIMER0:
          MOVWF     TIMER0     ;TIMER0 = 256 - 156 = 100
          BCF       INTCON,2   ;T0IF = 0.
ATTESA:   BTFSS     INTCON,2   ;Ciclo d'attesa
          GOTO      ATTESA     ;dell'evento T0IF = 1.
          MOVLW     00010000B  ;Inverti il livello del segnale
          XORWF     PORTA      ;in uscita dalla linea RA4.
          GOTO      CICLO
```

35°) Scrivere il programma di gestione di un microcontrollore PIC 16F84 che opera alla frequenza di 20 Mhz e che produce, sulla linea d'uscita RA4, un'onda quadra di frequenza pari a 500 Hz solo quando la linea d'ingresso RB0 è a livello basso.

36°) Scrivere il programma di gestione di un organo elettronico composto da un PIC 16F84, funzionante alla frequenza di 20 Mhz, la cui linea RA4, programmata in uscita, è collegata ad un altoparlante che a sua volta è collegato, tramite una resistenza di limitazione della corrente, al polo positivo dell'alimentazione del circuito, mentre ciascuna linea della porta B, programmata in ingresso, è collegata ad un pulsante che a sua volta è collegato a massa.

Soluzione senza l'uso delle interruzioni:

```
MAIN:     BSF       STATUS,RP0;Seleziona il banco 1.
          MOVLW     11111111B
          MOVWF     TRISB     ;PORTB = INPUT.
          MOVLW     00001111B
          MOVWF     TRISA     ;RA4 = OUTPUT.
          MOVLW     00000101B
          MOVWF     OPTION    ;Inizializza il registro OPTION.
          BCF       STATUS,RP0;Seleziona il banco 0.
          BSF       PORTA,4   ;Poni RA4 a livello alto.
          CLRF      TIMER0    ;Inizializza il registro TIMER0.
          CLRF      INTCON,2  ;Resetta TIMER0 interrupt flag.

CICLO:    BTFSS     INTCON,2  ;Ciclo d'attesa dell'evento
          GOTO      CICLO     ;TIMER0 overflow (T0IF = 1).
          CLRF      INTCON,2  ;Resetta TIMER0 interrupt flag.

          BTFSS     PORTB,7   ;Se RB7 è a livello basso allora
          GOTO      DO        ;emetti la nota "DO".
          BTFSS     PORTB,6   ;Se RB6 è a livello basso allora
          GOTO      RE        ;emetti la nota "RE".
          BTFSS     PORTB,5   ;Se RB5 è a livello basso allora
          GOTO      MI        ;emetti la nota "MI".
          BTFSS     PORTB,4   ;Se RB4 è a livello basso allora
          GOTO      FA        ;emetti la nota "FA".
          BTFSS     PORTB,3   ;Se RB3 è a livello basso allora
          GOTO      SOL       ;emetti la nota "SOL".
          BTFSS     PORTB,2   ;Se RB2 è a livello basso allora
          GOTO      LA        ;emetti la nota "LA".
          BTFSS     PORTB,1   ;Se RB1 è a livello basso allora
          GOTO      SI        ;emetti la nota "SI".

          BSF       PORTA,4   ;Poni RA4 a livello alto.
          BTFSC     PORTB,0   ;Se RB0 è a livello alto allora
          GOTO      CICLO     ;ritorna all'inizio del ciclo.
          CALL      RITARDO   ;Attendi la fine del rimbalzo.
          BTFSC     PORTB,0   ;Se RB0 è a livello alto allora
          GOTO      CICLO     ;ritorna all'inizio del ciclo.

          BSF       STATUS,RP0;Seleziona il banco 1.
          INCF      OPTION,W  ;Incrementa di uno modulo otto
          ANDLW     00000111B ;il moltiplicatore del PRESCALER
          MOVWF     OPTION    ;e aggiorna il registro OPTION.
          BCF       STATUS,RP0;Seleziona il banco 0.
```

```
ATTESA:   BTFSS     PORTB,0    ;Ciclo d'attesa del rilascio del
          GOTO      ATTESA     ;tasto collegato alla linea RB0.
          CALL      RITARDO    ;Attendi la fine del rimbalzo.
          BTFSS     PORTB,0    ;Se il rimbalzo non è terminato
          GOTO      ATTESA     ;Torna in attesa rilascio tasto
          GOTO      CICLO      ;altrimenti ritorna a CICLO.

DO:       MOVLW     00010000B ;Inverti il livello del segnale
          XORWF     PORTA      ;in uscita dalla linea RA4.
          MOVLW     107        ;Imposta nel TIMER0 il periodo
          MOVWF     TIMER0     ;della nota DO.
          GOTO      CICLO      ;Ritorna all'inizio del ciclo.

RE:       MOVLW     00010000B ;Inverti il livello del segnale
          XORWF     PORTA      ;in uscita dalla linea RA4.
          MOVLW     115        ;Imposta nel TIMER0 il periodo
          MOVWF     TIMER0     ;della nota RE.
          GOTO      CICLO      ;Ritorna all'inizio del ciclo.

MI:       MOVLW     00010000B ;Inverti il livello del segnale
          XORWF     PORTA      ;in uscita dalla linea RA4.
          MOVLW     ???        ;Imposta nel TIMER0 il periodo
          MOVWF     TIMER0     ;della nota MI.
          GOTO      CICLO      ;Ritorna all'inizio del ciclo.

FA:       MOVLW     00010000B ;Inverti il livello del segnale
          XORWF     PORTA      ;in uscita dalla linea RA4.
          MOVLW     ???        ;Imposta nel TIMER0 il periodo
          MOVWF     TIMER0     ;della nota FA.
          GOTO      CICLO      ;Ritorna all'inizio del ciclo.

SOL:      MOVLW     00010000B ;Inverti il livello del segnale
          XORWF     PORTA      ;in uscita dalla linea RA4.
          MOVLW     ???        ;Imposta nel TIMER0 il periodo
          MOVWF     TIMER0     ;della nota SOL.
          GOTO      CICLO      ;Ritorna all'inizio del ciclo.

LA:       MOVLW     00010000B ;Inverti il livello del segnale
          XORWF     PORTA      ;in uscita dalla linea RA4.
          MOVLW     167        ;Imposta nel TIMER0 il periodo
          MOVWF     TIMER0     ;della nota LA.
          GOTO      CICLO      ;Ritorna all'inizio del ciclo.

SI:       MOVLW     00010000B ;Inverti il livello del segnale
          XORWF     PORTA      ;in uscita dalla linea RA4.
          MOVLW     177        ;Imposta nel TIMER0 il periodo
          MOVWF     TIMER0     ;della nota SI.
          GOTO      CICLO      ;Ritorna all'inizio del ciclo.
```

Soluzione con l'uso delle interruzioni:

```
RESET:     ORG       0
           GOTO      MAIN
W_TEMP     EQU       0CH        ;Registro in cui salvare W.
STATUS_TEMP EQU      0DH        ;Registro in cui salvare STATUS.
ISR:       ORG       4          ;Indirizzo iniziale.
           MOVWF     W_TEMP     ;Salva il registro W.
           SWAPF     STATUS,W   ;Salva il registro STATUS.
           MOVWF     STATUS_TEMP
           BCF       STATUS,RP0;Seleziona il banco 0.
           BCF       INTCON,2   ;T0IF = 0.
           BTFSS     PORTB,7    ;Se RB7 è a livello basso allora
           GOTO      DO         ;emetti la nota "DO".
           BTFSS     PORTB,6    ;Se RB6 è a livello basso allora
           GOTO      RE         ;emetti la nota "RE".
           BTFSS     PORTB,5    ;Se RB5 è a livello basso allora
           GOTO      MI         ;emetti la nota "MI".
           BTFSS     PORTB,4    ;Se RB4 è a livello basso allora
           GOTO      FA         ;emetti la nota "FA".
           BTFSS     PORTB,3    ;Se RB3 è a livello basso allora
           GOTO      SOL        ;emetti la nota "SOL".
           BTFSS     PORTB,2    ;Se RB2 è a livello basso allora
           GOTO      LA         ;emetti la nota "LA".
           BTFSS     PORTB,1    ;Se RB1 è a livello basso allora
           GOTO      SI         ;emetti la nota "SI".
           BSF       PORTA,4    ;Poni RA4 a livello alto.

RIPRISTINO:                     ;Sequenza di ripristino:
           SWAPF     STATUS_TEMP,W
           MOVWF     STATUS     ;Ripristina il registro STATUS.
           SWAPF     W_TEMP
           SWAPF     W_TEMP,W   ;Ripristina il registro W.
           RETFIE               ;Ritorna al programma interrotto

DO:        MOVLW     00010000B ;Inverti il livello del segnale
           XORWF     PORTA      ;in uscita dalla linea RA4.
           MOVLW     107        ;Imposta nel TIMER0 il periodo
           MOVWF     TIMER0     ;della nota DO.
           GOTO      RIPRISTINO;Vai alla sequenza di ripristino

RE:        MOVLW     00010000B ;Inverti il livello del segnale
           XORWF     PORTA      ;in uscita dalla linea RA4.
           MOVLW     115        ;Imposta nel TIMER0 il periodo
           MOVWF     TIMER0     ;della nota RE.
           GOTO      RIPRISTINO;Vai alla sequenza di ripristino

MI:        MOVLW     00010000B ;Inverti il livello del segnale
           XORWF     PORTA      ;in uscita dalla linea RA4.
           MOVLW     ???        ;Imposta nel TIMER0 il periodo
           MOVWF     TIMER0     ;della nota MI.
           GOTO      RIPRISTINO;Vai alla sequenza di ripristino

FA:        MOVLW     00010000B ;Inverti il livello del segnale
```

```
            XORWF     PORTA     ;in uscita dalla linea RA4.
            MOVLW     ???       ;Imposta nel TIMER0 il periodo
            MOVWF     TIMER0    ;della nota FA.
            GOTO      RIPRISTINO;Vai alla sequenza di ripristino

SOL:        MOVLW     00010000B ;Inverti il livello del segnale
            XORWF     PORTA     ;in uscita dalla linea RA4.
            MOVLW     ???       ;Imposta nel TIMER0 il periodo
            MOVWF     TIMER0    ;della nota SOL.
            GOTO      RIPRISTINO;Vai alla sequenza di ripristino

LA:         MOVLW     00010000B ;Inverti il livello del segnale
            XORWF     PORTA     ;in uscita dalla linea RA4.
            MOVLW     167       ;Imposta nel TIMER0 il periodo
            MOVWF     TIMER0    ;della nota LA.
            GOTO      RIPRISTINO;Vai alla sequenza di ripristino

SI:         MOVLW     00010000B ;Inverti il livello del segnale
            XORWF     PORTA     ;in uscita dalla linea RA4.
            MOVLW     177       ;Imposta nel TIMER0 il periodo
            MOVWF     TIMER0    ;della nota SI.
            GOTO      RIPRISTINO;Vai alla sequenza di ripristino

MAIN:       BSF       STATUS,RP0;Seleziona il banco 1.
            MOVLW     11111111B
            MOVWF     TRISB     ;PORTB = INPUT.
            MOVLW     00001111B
            MOVWF     TRISA     ;RA4 = OUTPUT.
            MOVLW     00000101B
            MOVWF     OPTION    ;Inizializza il registro OPTION.
            BCF       STATUS,RP0;Seleziona il banco 0.
            BSF       PORTA,4   ;Poni RA4 a livello alto.
            MOVLW     10100000B ;Abilita le interruzioni
            MOVWF     INTCON    ;di tipo TIMER0 overflow.
CICLO:      BTFSC     PORTB,0   ;Se RB0 è a livello alto allora
            GOTO      CICLO     ;ritorna all'inizio del ciclo.
            CALL      RITARDO   ;Attendi la fine del rimbalzo.
            BTFSC     PORTB,0   ;Se RB0 è a livello alto allora
            GOTO      CICLO     ;ritorna all'inizio del ciclo.

            BSF       STATUS,RP0;Seleziona il banco 1.
            INCF      OPTION,W  ;Incrementa di uno modulo otto
            ANDLW     00000111B ;il moltiplicatore del PRESCALER
            MOVWF     OPTION    ;e aggiorna il registro OPTION.
            BCF       STATUS,RP0;Seleziona il banco 0.

ATTESA:     BTFSS     PORTB,0   ;Ciclo d'attesa del rilascio del
            GOTO      ATTESA    ;tasto collegato alla linea RB0.
            CALL      RITARDO   ;Attendi la fine del rimbalzo.
            BTFSS     PORTB,0   ;Se il rimbalzo non è terminato
            GOTO      ATTESA    ;Torna in attesa rilascio tasto
            GOTO      CICLO     ;altrimenti ritorna a CICLO.
            END                 ;Fine del programma.
```

37°) Scrivere il programma di gestione di un PIC 16F84 che produce in uscita sulle sette linee meno significative della porta B il codice a sette segmenti della cifra BCD presente in ingresso sulle quattro linee meno significative della porta A.

```
            ORG       0CH
CODICE      RES       10          ;Tabella dei codici a 7 segmenti
            ORG       0           ;delle cifre da 0 a 9.
INIZIO:     MOVLW     00111111B ;CODICE[0] = codice a 7 segmenti
            MOVWF     CODICE[0] ;della cifra 0.
            MOVLW     00000110B ;CODICE[1] = codice a 7 segmenti
            MOVWF     CODICE[1] ;della cifra 1.
            MOVLW     01011011B ;CODICE[2] = codice a 7 segmenti
            MOVWF     CODICE[2] ;della cifra 2.
            MOVLW     01001111B ;CODICE[3] = codice a 7 segmenti
            MOVWF     CODICE[3] ;della cifra 3.
            MOVLW     01100100B ;CODICE[4] = codice a 7 segmenti
            MOVWF     CODICE[4] ;della cifra 4.
            MOVLW     01101101B ;CODICE[5] = codice a 7 segmenti
            MOVWF     CODICE[5] ;della cifra 5.
            MOVLW     01111101B ;CODICE[6] = codice a 7 segmenti
            MOVWF     CODICE[6] ;della cifra 6.
            MOVLW     00000111B ;CODICE[7] = codice a 7 segmenti
            MOVWF     CODICE[7] ;della cifra 7.
            MOVLW     01111111B ;CODICE[8] = codice a 7 segmenti
            MOVWF     CODICE[8] ;della cifra 8.
            MOVLW     01101111B ;CODICE[9] = codice a 7 segmenti
            MOVWF     CODICE[9] ;della cifra 9.

            BSF       STATUS,RP0;Seleziona il banco 1.
            MOVLW     00011111B
            MOVWF     TRISA     ;PORTA = INPUT.
            CLRF      TRISB     ;PORTB = OUTPUT.
            BCF       STATUS,RP0;Seleziona il banco 0.

CICLO:      MOVF      PORTA,W   ;Lettura della porta A.
            ANDLW     0FH       ;W = codice BCD della cifra.
            ADDLW     CODICE
            MOVWF     FSR
            MOVF      INDF,W    ;W = CODICE[FSR + W].
            MOVWF     PORTB     ;PORTB = W
            GOTO      CICLO     ;Ritorna all'inizio del ciclo.
            END
```

38°) Un PIC 16F84 funziona alla frequenza di 20 MHz: la linea d'ingresso RA4 è collegata ad un pulsante normalmente aperto che a sua volta è collegato a massa. Le linee d'uscita RA0, RA1, RA2 ed RA3 sono collegate ai catodi di quattro display a LED a sette segmenti i cui anodi A, B, C, D, E ed F sono collegati nell'ordine alle linee d'uscita RB0, RB1, RB2, RB3, RB4, RB5 ed RB6 del microcontrollore. Scrivere un programma che visualizzi sui quattro display il numero di volte che il tasto è stato premuto a partire dall'istante di accensione del sistema.

```
ORG       0CH
CODICE    RES       10        ;Tabella dei codici a 7 segmenti
DISPLAY   RES       1         ;Indice del display attivo.
CIFRA     RES       4         ;codice BCD spacchettato del
                              ;numero da visualizzare.
W_TEMP    RES       1         ;Registro in cui salvare W.
STATUS_TEMP RES     1         ;Registro in cui salvare STATUS.
RESET:    ORG       0         ;Indirizzo di reset.
          GOTO      MAIN
ISR:      ORG       4         ;Indirizzo iniziale.
          MOVWF     W_TEMP    ;Salva il registro W.
          SWAPF     STATUS,W  ;Salva il registro STATUS.
          MOVWF     STATUS_TEMP

          MOVLW     61        ;Inizializza TIMER0 in modo che
          MOVWF     TIMER0    ;vada in overflow dopo 10 ms.
          BCF       INTCON,2  ;T0IF = 0.

          INCF      DISPLAY   ;Incrementa
          MOVLW     03H       ;DISPLAY
          ANDWF     DISPLAY   ;modulo 4.

          MOVF      DISPLAY,W ;W = contenuto di DISPLAY.
          ADDLW     CIFRA     ;W = indirizzo di CIFRA[DISPLAY]
          MOVWF     FSR       ;FSR = W.
          MOVF      INDF,W    ;W = contenuto di CIFRA[DISPLAY]
          ADDLW     CODICE    ;    + indirizzo di CODICE.
          MOVWF     FSR       ;W = codice a sette segmenti di
          MOVF      INDF,W    ;    CIFRA[DISPLAY].
          MOVWF     PORTB     ;PORTB = W.

          BSF       STATUS,C  ;Accendi la cifra successiva
          BTFSS     PORTA,3   ;del display a 7 segmenti.
          BCF       STATUS,C
          RLF       PORTA

          SWAPF     STATUS_TEMP,W
          MOVWF     STATUS    ;Ripristina il registro STATUS.
          SWAPF     W_TEMP
          SWAPF     W_TEMP,W  ;Ripristina il registro W.
          RETFIE              ;Ritorna al programma interrotto

MAIN:     MOVLW     00111111B ;CODICE[0] = codice a 7 segmenti
          MOVWF     CODICE[0] ;della cifra 0.
          MOVLW     00000110B ;CODICE[1] = codice a 7 segmenti
          MOVWF     CODICE[1] ;della cifra 1.
          MOVLW     01011011B ;CODICE[2] = codice a 7 segmenti
          MOVWF     CODICE[2] ;della cifra 2.
          MOVLW     01001111B ;CODICE[3] = codice a 7 segmenti
          MOVWF     CODICE[3] ;della cifra 3.
          MOVLW     01100100B ;CODICE[4] = codice a 7 segmenti
          MOVWF     CODICE[4] ;della cifra 4.
          MOVLW     01101101B ;CODICE[5] = codice a 7 segmenti
          MOVWF     CODICE[5] ;della cifra 5.
```

```
                MOVLW     01111101B ;CODICE[6] = codice a 7 segmenti
                MOVWF     CODICE[6] ;della cifra 6.
                MOVLW     00000111B ;CODICE[7] = codice a 7 segmenti
                MOVWF     CODICE[7] ;della cifra 7.
                MOVLW     01111111B ;CODICE[8] = codice a 7 segmenti
                MOVWF     CODICE[8] ;della cifra 8.
                MOVLW     01101111B ;CODICE[9] = codice a 7 segmenti
                MOVWF     CODICE[9] ;della cifra 9.
                CLRF      DISPLAY   ;E' attivo il display numero 0.
                CLRF      CIFRA[0]  ;Azzera tutte
                CLRF      CIFRA[1]  ;le cifre del
                CLRF      CIFRA[2]  ;numero da
                CLRF      CIFRA[3]  ;visualizzare.
                BSF       STATUS,RP0;Seleziona il banco 1.
                MOVLW     00010000B ;RA4 = INPUT,
                MOVWF     TRISA     ;RA0..3 = OUTPUT.
                CLRF      TRISB     ;PORTB = OUTPUT.
                MOVLW     00000111B ;Inizializza il registro OPTION
                MOVWF     OPTION    ;in modo che PRESCALER = 7.
                BCF       STATUS,5  ;Seleziona il banco 0.
                CLRF      PORTB     ;PORTB = 0.
                MOVLW     00001110B ;Accendi il display numero 0.
                MOVWF     PORTA
                MOVLW     61        ;Inizializza TIMER0 in modo che
                MOVWF     TIMER0    ;vada in overflow dopo 10 ms.
                MOVLW     10100000B ;Abilita le interruzioni
                MOVWF     INTCON    ;di tipo TIMER0 overflow.

CICLO:          BTFSC     PORTA,4   ;Se RA4 è a livello alto allora
                GOTO      CICLO     ;ritorna all'inizio del ciclo.
                CALL      RITARDO   ;Attendi la fine del rimbalzo.
                BTFSC     PORTA,4   ;Se RA4 è a livello alto allora
                GOTO      CICLO     ;ritorna all'inizio del ciclo.

                MOVF      CIFRA,W   ;FSR = indirizzo del numero da
                MOVWF     FSR       ;      visualizzare sul display.
                MOVLW     4         ;W = numero di cifre del numero.
                CALL      INC_NUM   ;Incrementa di uno il numero.
ATTESA:         BTFSS     PORTA,4   ;Ciclo d'attesa del rilascio del
                GOTO      ATTESA    ;tasto collegato alla linea RA4.
                CALL      RITARDO   ;Attendi la fine del rimbalzo.
                BTFSS     PORTA,4   ;Se il rimbalzo non è terminato
                GOTO      ATTESA    ;Torna in attesa rilascio tasto
                GOTO      CICLO     ;altrimenti ritorna a CICLO.
                END                 ;Fine del programma.
```

39°) Un PIC 16F84 funziona alla frequenza di 20 MHz: la linea d'ingresso RA4 è collegata ad un pulsante normalmente aperto che a sua volta è collegato a massa. Le linee d'uscita RA0, RA1, RA2 ed RA3 sono collegate ai catodi di quattro display a LED a sette segmenti i cui anodi A, B, C, D, E ed F sono collegati nell'ordine alle linee d'uscita RB0, RB1, RB2, RB3, RB4, RB5 ed RB6 del microcontrollore. Realizzare un cronometro digitale che visualizzi sui quattro display il numero di minuti e di secondi trascorsi dall'ultima pressione del pulsante, arrestando il conteggio all'istante della sua pressione successiva.

40°) Scrivere il programma di gestione di un microcontrollore PIC 16F84 che operi alla frequenza di 4 MHz e che produca, sulla linea d'uscita RA4, un'onda quadra asimmetrica di periodo T = 51,2 millisecondi, la cui prima parte a livello alto duri un numero di microsecondi pari a 200 volte il valore del numero binario di otto cifre che è rappresentato dai segnali presenti sulle otto linee d'ingresso della porta B.

Soluzione senza l'uso delle interruzioni:

```
CONTA     EQU       20H       ;Indirizzo del reg. contatore.
INIZIO:   BSF       STATUS,RP0;Seleziona il banco 1.
          MOVLW     00001111B
          MOVWF     TRISA     ;RA4 = OUTPUT.
          MOVLW     11111111B
          MOVWF     TRISB     ;PORTB = INPUT.
          MOVLW     00000000B
          MOVWF     OPTION    ;PRESCALER = 0 (divisore = 2).
          BCF       STATUS,RP0;Seleziona il banco 0.

          MOVLW     156       ;Inizializza TIMER0 in modo che
          MOVWF     TIMER0    ;vada in overflow dopo 200 µs.
          BCF       INTCON,2  ;Azzera il TIMER0 interrupt flag
          CLRF      CONTA     ;Registro contatore = 0.

CICLO:    MOVF      PORTB,W   ;W = PORTB.
          SUBWF     CONTA,W   ;W = CONTA - PORTB.
          BTFSS     STATUS,C  ;Se CONTA < PORTB (Carry = 0)
          BSF       PORTA,4   ;   RA4 = livello alto.
          BTFSC     STATUS,C  ;Se CONTA ≥ PORTB (Carry = 1)
          BCF       PORTA,4   ;   RA4 = livello basso.

ATTESA:   BTFSS     INTCON,2  ;Ciclo d'attesa dell'evento
          GOTO      ATTESA    ;TIMER0 overflow flag = 1.
          MOVLW     156       ;Inizializza TIMER0 in modo che
          MOVWF     TIMER0    ;vada in overflow dopo 200 µs.
          BCF       INTCON,2  ;Azzera il TIMER0 overflow flag.
          INCF      CONTA     ;Incrementa il reg. contatore.
          GOTO      CICLO     ;Ripeti il ciclo.
```

Soluzione con l'uso delle interruzioni:

```
CONTA     EQU       20H       ;Indirizzo del reg. contatore.

RESET:    ORG       0         ;Indirizzo di reset.
          GOTO      MAIN      ;Dopo il reset vai a MAIN.

ISR:      ORG       4         ;Indirizzo iniziale della ISR.
          MOVWF     W_TEMP    ;Salva il registro W.
          SWAPF     STATUS,W  ;Salva il registro STATUS.
          MOVWF     STATUS_TEMP

          MOVLW     156       ;Inizializza TIMER0 in modo che
          MOVWF     TIMER0    ;vada in overflow dopo 200 µs.
          BCF       INTCON,2  ;Azzera il TIMER0 overflow flag.
          INCF      CONTA     ;Incrementa il reg. contatore.
```

```
            SWAPF     STATUS_TEMP,W
            MOVWF     STATUS;Ripristina il registro STATUS.
            SWAPF     W_TEMP
            SWAPF     W_TEMP,W;Ripristina il registro W.
            RETFIE    ;Ritorna al programma interrotto.

MAIN:       BSF       STATUS,RP0;Seleziona il banco 1.
            MOVLW     00001111B
            MOVWF     TRISA     ;RA4 = OUTPUT.
            MOVLW     11111111B
            MOVWF     TRISB     ;PORTB = INPUT.
            MOVLW     00000000B
            MOVWF     OPTION    ;PRESCALER = 0 (divisore = 2).
            BCF       STATUS,RP0;Seleziona il banco 0.

            CLRF      CONTA     ;Registro contatore = 0.
            MOVLW     156       ;Inizializza TIMER0 in modo che
            MOVWF     TIMER0    ;in overflow dopo 200 µs.
            MOVLW     10100000B ;Abilita le interruzioni di tipo
            MOVWF     INTCON    ;TIMER0 overflow e poni T0IF = 0

CICLO:      MOVF      PORTB,W   ;W = PORTB.
            SUBWF     CONTA,W   ;W = CONTA - PORTB.
            BTFSS     STATUS,C  ;Se CONTA < PORTB (Carry = 0)
            BSF       PORTA,4   ;   RA4 = livello alto.
            BTFSC     STATUS,C  ;Se CONTA ≥ PORTB (Carry = 1)
            BCF       PORTA,4   ;   RA4 = livello basso.
            GOTO      CICLO     ;Ripeti il ciclo.
            END                 ;Fine del programma.
```

ESERCIZI IN LINGUAGGIO JAVA

```
/**
 * Prova1: dimostrazione del funzionamento dei Thread.
 */
public class MyThread extends Thread
{
    private String message;

    public MyThread(String m)
    {
        message = m;
    }
    /**
     * Metodo run:
     *
     * Ripete 100 volte le seguenti operazioni:
     * 1. visualizza il nome di questo Thread seguito dal
          numero del messaggio.
     * 2. Attende per un periodo di 100 millisecondi.
     */
    public void run()
    {
        for (int i = 0; i < 100; i++)
        {   System.out.println (message + i);
            try {
                sleep(100);
            } catch (Exception e) {}
        }
    }
    /**
     * Metodo di collaudo:
     *
     * Crea tre thread di classe MyThread
     * e nome rispettivamente "A" "B" e "C"
     * e li attiva ad intervalli di 20 ms.
     * l'uno dall'altro.
     */
    public static void main(String[] args) throws Exception
    {
        Thread A = new MyThread("A");
        Thread B = new MyThread("       B");
        Thread C = new MyThread("               C");

        A.start();
        sleep(20);
        B.start();
        sleep(20);
        C.start();
    }
}
```

```
/**
 * Prova2: dimostrazione del funzionamento dei Thread.
 */
public class MyThread extends Thread
{
    private static long millisec;
    private String message;
    /**
     * Costruttore della classe:
     * Memorizza il parametro di chiamata nel
     * campo "message" ed attiva il thread.
     */
    public MyThread(String m)
    {
        message = m;
        start();
    }
    /**
     * Metodo run:
     * Ripete 100 volte le seguenti operazioni:
     *  1. Visualizza la stringa contenuta nel campo
     *     "message" seguito dal numero del messaggio.
     *  2. Attende per un periodo di "millisec" ms.
     */
    public void run()
    {
        for (int i = 0; i < 100; i++)
        {   System.out.println (message + i);
            try {sleep(millisec);
            } catch (Exception e) {}
        }
    }
    /**
     * Metodo di collaudo:
     * Crea un oggetto di classe MyThread per
     * ogni argomento di chiamata del metodo "main",
     * passando tale argomento come parametro del
     * costruttore.
     * Tutti questi thread sono attivati ad intervalli
     * di 100 ms. l'uno dall'altro.
     */
    public static void main(String[] args) throws Exception
    {
        millisec = args.length * 100;
        for (int i = 0; i < args.length; i++)
        {   String name = "";
            for (int j = 0; j < i; j++)
                name = name + "\t";
            new MyThread(name + args[i]);
            sleep(100);
        }
    }
}
```

```
/**
 * Prova3: esempio di corsa critica e sua soluzione con l'uso di
 *         metodi sincronizzati.
 *
 */
public interface Number
{
    public int  get();      //Restituisce il valore di un oggetto.
    public void set(int n); //Imposta un nuovo valore dell'oggetto
    public void inc();      //Incrementa il valore dell'oggetto.
}
/*
 * Implementazione dell'interfaccia Number
 * per mezzo di numeri interi.
 */
public class IntNumber implements Number
{
    private int number;
    public IntNumber()
    {  number = 0;
    }
    public int get()
    {   return number;
    }
    public void set(int n)
    {   number = n;
    }
    public synchronized void inc()
    {    number +=1;
    }
}
/*
 * Implementazione dell'interfaccia Number
 * per mezzo di stringhe di caratteri.
 */
public class StringNumber implements Number
{
    private String number;
    public StringNumber()
    {  number = Integer.toString(0);
    }
    public int get()
    {   return Integer.parseInt(number);
    }
    public void set(int n)
    {   number = Integer.toString(n);
    }
    public synchronized void inc()
    {   int n = get();
        n +=1;
        set(n);
    }
}
```

```
public class Incrementer1 extends Thread
{
    Number num;

    public Incrementer1(Number n)
    {
        num = n;
    }

    public void run()
    {
        for (int i = 0; i < 1000; i++)
            num.inc();
    }
    /**
     * Metodo di collaudo della classe Incrementer1:
     * Dopo aver creato un oggetto di classe StringNumber,
     * lo passa come parametro ai costruttori di due thread
     * di classe Incrementer1 che ne incrementano il valore
     * contemporaneamente. Dopo la terminazione dei due
     * thread, viene visualizzato il valore del numero.
     */
    public static void main(String[] args) throws Exception
    {
        Number n = new StringNumber();
        Thread a = new Incrementer1(n);
        Thread b = new Incrementer1(n);

        a.start();
        b.start();

        a.join();
        b.join();
        System.out.println("Valore finale del numero: "+n.get());
    }
}
```

```
public class Incrementer2 extends Thread
{
    Number num;

    public Incrementer2(Number n)
    {
        num = n;
    }

    public void run()
    {
        for (int i = 0; i < 1000; i++)
            num.inc();
    }
    /**
     * Metodo di collaudo della classe Incrementer2:
     * Dopo aver creato un oggetto di classe IntNumber,
     * lo passa come parametro ai costruttori di due thread
     * di classe Incrementer2 che ne incrementano il valore
     * contemporaneamente. Dopo la terminazione dei due
     * thread, viene visualizzato il valore del numero.
     */
    public static void main(String[] args) throws Exception
    {
        Number n = new IntNumber();
        Thread a = new Incrementer2(n);
        Thread b = new Incrementer2(n);

        a.start();
        b.start();

        a.join();
        b.join();
        System.out.println("Valore finale del numero: "+n.get());
    }
}
```

```
/**
 * Prova4: scambio di messaggi tra due thread
 *         per mezzo di un buffer di stringhe
 *         senza l'uso dei metodi wait e notify.
 */
public interface Buffer
{
    public String get();    //Preleva un messaggio dal buffer;
                            //se il buffer è vuoto, restituisce
                            //il valore "null".
    public boolean put(String m);
                            //Inserisce un messaggio nel buffer;
                            //se il buffer è pieno, restituisce
                            //il valore "false".
}
/**
 * Implementazione dell'interfaccia buffer
 * tramite un buffer ad una sola posizione.
 */
public class SimpleBuffer implements Buffer
{
    private String msg;
    public SimpleBuffer()
    {   msg = null;
    }
    public synchronized String get()
    {
        String temp = msg;
        msg = null;
        return temp;
    }
    public synchronized boolean put(String m)
    {
        if (msg != null) return false;
        msg = m;
        return true;
    }
    /**
     * Metodo di collaudo:
     *  Il primo argomento rappresenta il tempo necessario
     *  al thread produttore per creare un nuovo messaggio.
     *  Il secondo argomento rappresenta il tempo necessario
     *  al thread consumatore per prelevare un messaggio.
     */
    public static void main(String[] args)
    {
        Buffer b = new SimpleBuffer();
        Thread p = new Producer(b, Long.parseLong(args[0]));
        Thread c = new Consumer(b, Long.parseLong(args[1]));

        c.start();
        p.start();
    }
}
```

```
/**
 * Implementazione dell'interfaccia buffer
 * tramite un buffer a più posizioni.
 */
public class MultiBuffer implements Buffer
{
    private String[] msg;
    private int first, num;

    public MultiBuffer(int capacity)
    {
        msg = new String[capacity];
        first = num = 0;
    }
    public synchronized String get()
    {
        if (num == 0) return null;
        String temp = msg[first];
        first = (first + 1) % msg.length;
        num -= 1;
        return temp;
    }
    public synchronized boolean put(String m)
    {
        if (num == msg.length) return false;
        int next = (first + num) % msg.length;
        msg[next] = m;
        num += 1;
        return true;
    }
    /**
     * Metodo di collaudo:
     *  Il primo argomento rappresenta il tempo necessario
     *  al thread produttore per creare un nuovo messaggio.
     *  Il secondo argomento rappresenta il tempo necessario
     *  al thread consumatore per prelevare un messaggio.
     *  Il terzo argomento rappresenta la capacità del buffer.
     */
    public static void main(String[] args)
    {
        Buffer b = new MultiBuffer(Integer.parseInt(args[2]));
        Thread p = new Producer(b, Long.parseLong(args[0]));
        Thread c = new Consumer(b, Long.parseLong(args[1]));

        c.start();
        p.start();
    }
}
```

```
/**
 * Thread produttore di messaggi
 */
public class Producer extends Thread
{
    private Buffer buffer;
    private long delay;

    public Producer(Buffer b, long d)
    {
        buffer = b;
        delay = d;
    }
    public void run()
    {
        for (int i = 0; i < 100; i++)
            try {while (!buffer.put("Messaggio numero " + i))
yield();
                sleep(delay);
            } catch (Exception e) {}
        while (!buffer.put("")) yield();
        System.out.println("Fine del thread produttore!");
    }
}
/**
 * Thread consumatore di messaggi
 */
public class Consumer extends Thread
{
    private Buffer buffer;
    private long delay;

    public Consumer(Buffer b, long d)
    {
       buffer = b;
       delay = d;
    }
    public void run()
    {
        String m;
        do {try {while ((m = buffer.get()) == null) yield();
                sleep(delay);
            } catch (Exception e) {m = null;};
            System.out.println(m);
        } while (m.length() > 0);
        System.out.println("Fine del thread consumatore!");
    }
}
```

```
/**
 * Prova5: scambio di messaggi tra due thread
 *         per mezzo di un buffer di stringhe
 *         con l'uso dei metodi wait e notify.
 */
public interface Buffer
{
    public String get() throws Exception;
                            //Preleva un messaggio dal buffer;
                            //se il buffer è vuoto il thread
                            //chiamante viene bloccato.
    public void put(String m) throws Exception;
                            //Inserisce un messaggio nel buffer;
                            //se il buffer è pieno il thread
                            //chiamante viene bloccato.
}
/**
 * Implementazione dell'interfaccia buffer
 * tramite un buffer ad una sola posizione.
 */
public class SimpleBuffer implements Buffer
{
    private String msg;
    public SimpleBuffer()
    {   msg = null;
    }
    public synchronized String get() throws Exception
    {   while (msg == null) wait();
        String temp = msg;
        msg = null;
        notify();
        return temp;
    }
    public synchronized void put(String m) throws Exception
    {   while (msg != null) wait();
        msg = m;
        notify();
    }
    /**
     * Metodo di collaudo:
     *  Il primo argomento rappresenta il tempo necessario
     *  al thread produttore per creare un nuovo messaggio.
     *  Il secondo argomento rappresenta il tempo necessario
     *  al thread consumatore per prelevare un messaggio.
     */
    public static void main(String[] args)
    {   Buffer b = new SimpleBuffer();
        Thread p = new Producer(b, Long.parseLong(args[0]));
        Thread c = new Consumer(b, Long.parseLong(args[1]));

        c.start();
        p.start();
    }
}
```

```
/**
 * Implementazione dell'interfaccia buffer
 * tramite un buffer a più posizioni.
 */
public class MultiBuffer implements Buffer
{
    private String[] msg;
    private int first, num;

    public MultiBuffer(int capacity)
    {
        msg = new String[capacity];
        first = num = 0;
    }
    public synchronized String get() throws Exception
    {
        while (num == 0) wait();
        String temp = msg[first];
        first = (first + 1) % msg.length;
        num -= 1;
        notify();
        return temp;
    }
    public synchronized void put(String m) throws Exception
    {
        while (num == msg.length) wait();
        int next = (first + num) % msg.length;
        msg[next] = m;
        num += 1;
        notify();
    }
    /**
     * Metodo di collaudo:
     *  Il primo argomento rappresenta il tempo necessario
     *  al thread produttore per creare un nuovo messaggio.
     *  Il secondo argomento rappresenta il tempo necessario
     *  al thread consumatore per prelevare un messaggio.
     *  Il terzo argomento rappresenta la capacità del buffer.
     */
    public static void main(String[] args)
    {
        Buffer b = new MultiBuffer(Integer.parseInt(args[2]));
        Thread p = new Producer(b, Long.parseLong(args[0]));
        Thread c = new Consumer(b, Long.parseLong(args[1]));

        c.start();
        p.start();
    }
}
```

```
/**
 * Thread produttore di messaggi
 */
public class Producer extends Thread
{
    private Buffer buffer;
    private long delay;

    public Producer(Buffer b, long d)
    {
        buffer = b;
        delay = d;
    }
    public void run()
    {
        for (int i = 0; i < 100; i++)
            try {buffer.put("Messaggio numero " + i);
                sleep(delay);
            } catch (Exception e) {}
        try {buffer.put("");
        } catch (Exception e) {}
        System.out.println("Fine del thread produttore!");
    }
}
/**
 * Thread consumatore di messaggi
 */
public class Consumer extends Thread
{
    private Buffer buffer;
    private long delay;

    public Consumer(Buffer b, long d)
    {
       buffer = b;
       delay = d;
    }
    public void run()
    {
        String m;
        do {try {m = buffer.get();
                sleep(delay);
            } catch (Exception e) {m = null;};
            System.out.println(m);
        } while (m.length() > 0);
        System.out.println("Fine del thread consumatore!");
    }
}
```

```
/**
 * Prova6: scambio di dati tra due thread
 *         per mezzo di un pipe di caratteri
 *         con l'uso dei metodi wait e notify.
 */
public interface Pipe
{
    public char get();
                            //Preleva un carattere dal pipe;
                            //se il pipe è vuoto il thread
                            //chiamante viene bloccato.
    public void put(char c);
                            //Inserisce un carattere nel pipe;
                            //se il pipe è pieno il thread
                            //chiamante viene bloccato.
    public void read(char[] block);
                            //Preleva un blocco di caratteri
                            //dal pipe copiandoli nel vettore
                            //di caratteri "block".
    public void write(char[] block);
                            //Inserisce un blocco di caratteri
                            //nel pipe copiandoli dal vettore
                            //di caratteri "block".
}
```

```
/**
 * Implementazione dell'interfaccia pipe
 * tramite un buffer ad una sola posizione.
 */
public class SimplePipe implements Pipe
{
    private char elem;
    private boolean empty;

    public SimplePipe()
    {   empty = true;
    }
    public synchronized char get()
    {   while (empty)
            try {wait();
            } catch (Exception e) {}
        empty = true;
        notify ();
        return elem;
    }
    public synchronized void put(char c)
    {   while (!empty)
            try {wait();
            } catch (Exception e) {}
        elem = c;
        empty = false;
        notify();
    }
    public synchronized void read(char[] block)
    {   for (int i = 0; i < block.length; i++)
            block[i] = get();
    }
    public synchronized void write(char[] block)
    {   for (int i = 0; i < block.length; i++)
            put(block[i]);
    }
    /**
     * Metodo di collaudo:
     *  Il primo argomento rappresenta il blocco di caratteri
     *  che il thread produttore invia continuamente al pipe.
     *  Il secondo argomento rappresenta la lunghezza del
     *  blocco di caratteri che il thread consumatore preleva
     *  continuamente dal pipe e visualizza sullo schermo.
     */
    public static void main(String[] args)
    {
        Pipe pipe = new SimplePipe();
        Thread p = new Producer(pipe, args[0]);
        Thread c = new Consumer(pipe, args[1]);

        p.start();
        c.start();
    }
}
```

```
/**
 * Implementazione dell'interfaccia pipe
 * tramite un buffer multiplo.
 */
public class MultiPipe implements Pipe
{
    private char[] elem;
    private int first, num;

    public MultiPipe(int capacity)
    {   num = 0;
        elem = new char[capacity];
    }
    public synchronized char get()
    {   while (num == 0)
            try {wait();
            } catch (Exception e) {}
        char aux = elem[first];
        first = (first + 1) % elem.length;
        num -= 1;
        notify ();
        return aux;
    }
    public synchronized void put(char c)
    {   while (num == elem.length)
            try {wait();
            } catch (Exception e) {}
        elem[(first + num) % elem.length] = c;
        num += 1;
        notify();
    }
    public synchronized void read(char[] block)
    {   for (int i = 0; i < block.length; i++) block[i] = get();
    }
    public synchronized void write(char[] block)
    {   for (int i = 0; i < block.length; i++) put(block[i]);
    }
    /**
     * Metodo di collaudo:
     *  Il primo argomento rappresenta il blocco di caratteri
     *  che il thread produttore invia continuamente al pipe.
     *  Il secondo argomento rappresenta la lunghezza del
     *  blocco di caratteri che il thread consumatore preleva
     *  continuamente dal pipe e visualizza sullo schermo.
     *  Il terzo argomento rappresenta la capacità del pipe.
     */
    public static void main(String[] args)
    {   Pipe pipe = new MultiPipe(Integer.parseInt(args[2]));
        Thread p = new Producer(pipe, args[0]);
        Thread c = new Consumer(pipe, args[1]);
        p.start();
        c.start();
    }
}
```

```
/**
 * Thread produttore dei dati
 */
public class Producer extends Thread
{
    private Pipe pipe;
    private String msg;

    public Producer(Pipe p, String m)
    {
        pipe = p;
        msg = m;
    }
    public void run()
    {
        while (true)
            pipe.write(msg.toCharArray());
    }
}
/**
 * Thread consumatore dei dati
 */
public class Consumer extends Thread
{
    private Pipe pipe;
    private char[] msg;

    public Consumer(Pipe p, String l)
    {
        pipe = p;
        msg = new char[Integer.parseInt(l)];
    }
    public void run()
    {
        while (true)
        {   pipe.read(msg);
            System.out.println(msg);
        }
    }
}
```

```
import java.util.Random;
/**
 * Prova7:
 * Problema dei lettori-scrittori: un insieme di thread accede ad
 * una risorsa comune sia in lettura che in scrittura.
 * L'accesso in scrittura può essere fatto da un solo thread alla
 * volta. L'accesso in lettura è consentito contemporaneamente a
 * più thread solo se nessun altro sta accedendo in scrittura.
 */
public class Resource
{
    private int readers, writers;
    public Resource()
    {   readers = writers = 0;
    }
    public synchronized void readStart() throws Exception
    {   while (writers != 0) wait();
        readers += 1;
        System.out.println("Lettori = " + readers +
                           "\tScrittori = " + writers);
    }
    public synchronized void readStop()
    {   readers -= 1;
        System.out.println("Lettori = " + readers +
                           "\tScrittori = " + writers);
        if (readers == 0)
            notify();
    }
    public synchronized void writeStart() throws Exception
    {   while (readers + writers > 0) wait();
        writers += 1;
        System.out.println("Lettori = " + readers +
                          "\tScrittori = " + writers);
    }
    public synchronized void writeStop()
    {    writers -= 1;
        System.out.println("Lettori = " + readers +
                           "\tScrittori = " + writers);
        notifyAll();
    }
    /**
     * Metodo di collaudo:
     * Il primo argomento rappresenta il numero di thread lettori.
     * Il secondo argomento il numero di thread scrittori.
     */
    public static void main(String[] args)
    {
        Resource r = new Resource();
        Random   n = new Random();
        for (int i = 0; i < Integer.parseInt(args[0]); i++)
            new Reader(r, n.nextInt(1000), n.nextInt(9000));
        for (int i = 0; i < Integer.parseInt(args[1]); i++)
            new Writer(r, n.nextInt(1000), n.nextInt(9000));
    }
```

```
/**
 * Thread lettore:
 */
public class Reader extends Thread
{
    private Resource resource;
    private long readTime, noReadTime;
    public Reader(Resource r, long rt, long nrt)
    {
        resource = r;
        readTime = rt;
        noReadTime = nrt;
        start();
    }
    public void run()
    {
        while (true)
        {   try {
                sleep(noReadTime);
                resource.readStart();
                sleep(readTime);
                resource.readStop();
            } catch (Exception e) {}
        }
    }
}
/**
 * Thread scrittore:
 */
public class Writer extends Thread
{
    private Resource resource;
    private long writeTime, noWriteTime;

    public Writer(Resource r, long wt, long nwt)
    {
        resource = r;
        writeTime = wt;
        noWriteTime = nwt;
        start();
    }
    public void run()
    {
        while (true)
        {   try {
                sleep(noWriteTime);
                resource.writeStart();
                sleep(writeTime);
                resource.writeStop();
            } catch (Exception e) {}
        }
    }
}
```

```
/**
 * Prova8: problema del "barbiere che dorme".
 *
 * Un stazione di servizio possiede un certo numero di thread
 * server che eseguono i servizi richiesti da altri thread di
 * tipo client che arrivano nella stazione in continuazione.
 * Quando arriva un nuovo client ma non ci sono server liberi,
 * il client aspetta il suo turno. Se però la capacità della
 * stazione di ospitare client è esaurita, i nuovi client
 * che arrivano non possono entrare all'interno della stazione.
 */
public class Station
{
    private int chairs, clients, servers;
    public Station(int c)
    {
        chairs = c;
        clients = servers = 0;
    }
    public synchronized void startClient() throws Exception
    {
        if (clients == chairs)
        {   System.out.println("Richiesta rifiutata!");
            return;
        }
        clients += 1;
        System.out.println("Clienti in attesa = " + clients);
        notifyAll();
        while (servers == 0)
            wait();
        servers -= 1;
    }
    public synchronized void startService() throws Exception
    {
        servers += 1;
        notifyAll();
        while (clients == 0) wait();
        clients -= 1;
        System.out.println("Clienti in attesa = " + clients);
    }
    /**
     * Metodo di collaudo:
     * Il primo argomento rappresenta la capacità della stazione.
     * Il secondo rappresenta il tempo di servizio delle richieste
     * Il terzo rappresenta il tempo massimo tra l'arrivo di due
     * nuovi tread client.
     */
    public static void main(String[] args)
    {
        Station s = new Station(Integer.parseInt(args[0]));
        new Server(s, Integer.parseInt(args[1]));
        new ClientSource(s, Integer.parseInt(args[2]));
    }
}
```

```
/**
 * Thread server:
 */
public class Server extends Thread
{
    private Station shop;
    private long serviceTime;

    public Server(Station s, long st)
    {
        shop = s;
        serviceTime = st;
        start();
    }
    public void run()
    {
        while (true)
        {   try {
                shop.startService();
                sleep(serviceTime);
            } catch (Exception e) {}
        }
    }
}
import java.util.Random;
/**
 * Thread che genera i thread client:
 */
public class ClientSource extends Thread
{
    private Station station;
    private int waitTime;
    private Random rand;

    public ClientSource(Station s, int wt)
    {
        station = s;
        waitTime = wt;
        rand = new Random();
        start();
    }
    public void run()
    {
        for (int i = 0; i < 100; i++)
        {   try {
                new Client(station);
                sleep(rand.nextInt(waitTime));
            } catch (Exception e) {}
        }
    }
}
```

```
/**
 * Thread client:
 */
public class Client extends Thread
{
    private Station station;

    public Client(Station s)
    {
        station = s;
        start();
    }
    public void run()
    {
        try {
            station.startClient();
        } catch (Exception e) {}
    }
}
```

```
/**
 * Prova9: gestione di un insieme di risorse dello stesso tipo.
 */
public class Resource
{
    private boolean[] isUsed;

    public Resource(int n)
    {
        isUsed = new boolean[n];
        for (int i = 0; i < n; i++)
            isUsed[i] = false;
    }
    public synchronized int request()
    {
        while (true)
        {   for (int i = 0; i < isUsed.length; i++)
                if (!isUsed[i])
                {   isUsed[i] = true;
                    System.out.println("Assegnata risorsa " + i);
                    return i;
                }
            try {wait();
            }   catch (Exception e) {};
        }
    }
    public synchronized void release(int i)
    {
        System.out.println("\t\t\tRilasciata risorsa " + i);
        isUsed[i] = false;
        notify();
    }
    /**
     * Metodo di collaudo:
     * Il primo argomento rappresenta il numero di risorse
     * disponibili.
     * Gli argomenti successivi rappresentano i tempi di utilizzo
     * e di non utilizzo delle risorse da parte di ciascuno dei
     * thread utenti espresse in millisecondi.
     */
    public static void main(String[] args)
    {
        Resource r = new Resource(Integer.parseInt(args[0]));

        for (int i = 1; i < args.length; i++)
            new User(r, Long.parseLong(args[i]));
    }
}
```

```
/**
 * Thread utente:
 */
public class User extends Thread
{
    private Resource resource;
    private long time;

    public User(Resource r, long t)
    {
        resource = r;
        time = t;
        start();
    }
    public void run()
    {
        for (int i = 0; i < 10; i++) try
        {   sleep(time);
            int n = resource.request();
            sleep(time);
            resource.release(n);
        }
        catch (Exception e) {}
    }
}
```

```
/**
 * Prova10: problema dei filosofi a cena.
 */
public class Resource
{
    private int number;
    private boolean isUsed;

    public Resource(int n)
    {
        number = n;
        isUsed = false;
    }
    public synchronized int request()
    {
        while (isUsed) try
        {   wait();
        }
        catch (Exception e) {}
        isUsed = true;
        return number;
    }
    public synchronized int release()
    {
        isUsed = false;
        notifyAll();
        return number;
    }
    /**
     * Metodo di collaudo:
     * Il primo argomento rappresenta sia numero di risorse
     * disponibili, sia il numero di utenti di tali risorse.
     */
    public static void main(String[] args)
    {
        int n = Integer.parseInt(args[0]);
        Resource[] r = new Resource[n];
        Thread[] u = new User[n];

        for (int i = 0; i < n; i++)
            r[i] = new Resource(i);

        for (int i = 0; i < n; i++)
            u[i] = new User(r[i], r[(i + 1) % n]);
    }
}
```

```
import java.util.Random;
/**
 * Thread utente:
 */
public class User extends Thread
{
    private Resource left, right;
    private Random rand;

    public User(Resource l, Resource r)
    {
        left = l;
        right = r;
        rand = new Random();
        start();
    }
    public void run()
    {
        for (int i = 0; i < 100; i++) try
        {   sleep(rand.nextInt(100));
            System.out.println("Assegnata la risorsa  numero " +
                               left.request()  + " a " + getName());
            sleep(rand.nextInt(100));
            System.out.println("Assegnata la risorsa  numero " +
                               right.request() + " a " + getName());
            sleep(rand.nextInt(800));
            System.out.println("Rilasciata la risorsa numero " +
                               left.release() + " da " + getName());
            System.out.println("Rilasciata la risorsa numero " +
                               right.release()+ " da " + getName());
        }
        catch (Exception e) {}
        System.out.println("\t\t\t\t" + getName() + "terminato!");
    }
}
```

```
/**
 * Prova11: verifica delle priorità dei thread.
 */
public class Priority extends Thread
{
    public void run()
    {
        System.out.println("La priorità di questo  thread è " +
                                                    getPriority());
        System.out.println("La massima priorità possibile è " +
                            getThreadGroup().getMaxPriority());
    }
    public static void main(String[] args)
    {
        System.out.println("La priorità minima  di un thread è " +
                                                     MIN_PRIORITY);
        System.out.println("La priorità massima di un thread è " +
                                                     MAX_PRIORITY);
        System.out.println("La priorità default di un thread è " +
                                                    NORM_PRIORITY);
        Thread t = new Priority();
        t.start();
    }
}


/**
 * Prova12: creazione di più thread con differenti priorità.
 */
public class TestPriority extends Thread
{
    public void run()
    {
        for (int conta = 1; conta <= 4000000; conta++)
            if (conta % 500000 == 0)
                System.out.println(getName() + ":\tPriorità = " +
                        getPriority() + ":\tContatore = " + conta);
    }
    /**
     * Metodo di collaudo:
     *
     * Per ogni argomento di questo metodo viene creato un thread
     * e gli viene assegnata una priorità pari al valore di tale
     * argomento.
     */
    public static void main(String[] args)
    {
        for (int i = 0; i < args.length; i++)
        {   Thread t = new TestPriority();
            t.setPriority(Integer.parseInt(args[i]));
            t.start();
        }
    }
}
```