Midterm Solutions

1.

```
#define PAIN (volatile unsigned char *) 0xFFFFFFF0
#define PAOUT (volatile unsigned char *) 0xFFFFFFF1
#define PADIR (volatile unsigned char *) 0xFFFFFFF2
#define PBOUT (volatile unsigned char *) 0xFFFFFFF4
#define PBDIR (volatile unsigned char *) 0xFFFFFFF5
#define PSTAT (volatile unsigned char *) 0xFFFFFFF6
#define CNTM (volatile unsigned int *) 0xFFFFFFD0
#define CTCON (volatile unsigned char *) 0xFFFFFFD8
#define CTSTAT (volatile unsigned char *) 0xFFFFFFD9
#define IVECT (volatile unsigned int *) (0x20)

interrupt void intserv();

volatile unsigned char digit1 = 0;        /* DIGIT1 for display */
volatile unsigned char digit2 = 0;        /* DIGIT2 for display */
volatile unsigned char led1 = 0;          /* LED1 initially on */
volatile unsigned char led2 = 1;          /* LED2 initially off */

int main() {
  *PADIR = 0x4F;                          /* Set Port A direction */
  *PBDIR = 0xF1;                          /* Set Port B direction */
  *CTCON = 0x2;                           /* Stop Timer (if running) */
  *CTSTAT = 0x0;                          /* Clear "Reached 0" flag */
  *CNTM = 100000000;                      /* Initialize: 1-s timeout */
  *IVECT = (unsigned int *) &intserv;     /* Set interrupt vector */
  asm("MoveControl PSR,#0x40");           /* CPU responds to IRQ */
  *CTCON = 0x11;                          /* Start Timer, enable IRQ */
  while (1) {
    *PAOUT = (led2 << 6) | digit2;  /* Update LED2, same DIGIT2 */
    *PBOUT = (digit1 << 4) | led1;  /* Update LED1, same DIGIT1 */
    while ((*PAIN & 0x80) != 0);          /* Wait for SW press */
    while ((*PAIN & 0x80) == 0);          /* Wait for SW release */
    if (led1 == 0) {led1 = 1; led2 = 0;}  /* Swap LED1/LED2 states */
    else {led2 = 1; led1 = 0;}
  }
  exit(0);
}

interrupt void intserv() {
  *CTSTAT = 0x0;              /* Clear "Reached 0" flag */
  if (led1 == 0) {
    digit1 = (digit1 + 1)%10;       /* Increment DIGIT1 */
    *PBOUT = (digit1 << 4) | led1;  /* Update DIGIT1, same LED1 */
  }
  if (led2 == 0) {
    digit2 = (digit2 + 1)%10;       /* Increment DIGIT1 */
    *PAOUT = (led2 << 6) | digit2;  /* Update DIGIT2, same LED2 */
  }
}
```

2.

The LCM (least common multiple) of all four periods is 90, i.e., we only need to determine our EDF schedule in the time interval **[0, 90)**, after which it is repeated.

RM task priorities are: 1/30 for T1; 1/30 for T2; 1/45 for T3; 1/90 for T4.
EDF task priorities are: (1/20, 1/50, 1/80) for T1 arriving at (0, 30, 60); (1/30, 1/60, 1/90) for T2 arriving at (0, 30, 60); (1/25, 1/70) for T3 arriving at (0, 45); (1/80) for T4 arriving at (0).

RM Schedule	EDF Schedule
t=0: T1	t=0: T1
t=5: T2	t=5: T3
t=10: T3	t=20: T2
t=25: T4	t=25: T4
t=30: T1 (T4 preempted)	t=30: T1 (T4 preempted)
t=35: T2	t=35: T2
t=40: T4	t=40: T4
t=45: T3 (T4 preempted)	t=45: T3 (T4 preempted)
t=60: T1	t=60: T1
t=65: T2	t=65: T4
t=70: T4	t=70: T2
t=75: Idle	t=75: Idle
t=90: Repeat…	t=90: Repeat…

3.