```c
//Description:
//
//
//This example is using the timer1 and interrupt to make shifting LED at every second.
//The timer1 register(TMR1H:TMR1L) has 16 bits and we use the internal clock as clock source.
//Suppose we choose the Prescaler as 1:8.
//
//The timer increment frequency = (Fosc/4)*(1/8)= (48MHz/4)*(1/8)= 1.5MHz.
//This time, we want to generate one interrupt every 20ms.Then,
//the amount of the timer register value for 20 ms  = (20*e^-3)*(1.5*e^6) = 30000.
//
//The timer0 starting value = 0xFFFF-30000 =65535-30000 = 35535 =0x8ACF.
//Now we know every 20ms, one interrupt can be generated. As we want every 1 second, LED can
shift once.
//Then we can count the interrupts, if there have 50 interrupts been generated, we can get 1
second.
//
//Last, we make the LED shift during the interrupt service routine, and count the interrupt from
0 again.



#include <p18f4550.h>
#include <timers.h>       // Include the timer library

void timer1_isr(void);    //Interrupt service routine prototype
int i=0;                  //Interrupt counter

//Always include this code, it’s necessary when using a bootloader
extern void _startup (void);
#pragma code _RESET_INTERRUPT_VECTOR = 0x000800
void _reset (void)
{
_asm goto _startup _endasm
}
#pragma code
#pragma code _HIGH_INTERRUPT_VECTOR = 0x000808
 void high_ISR (void)
 {
  //Pre: The interrupt priority is defined as high and enabled.The high priority interrupt
service routine is called
  //Post: Execute the timer1_isr function.

 _asm goto timer1_isr _endasm  //when the high priority interrupt is excuted, go to timer1
interrupt service routine

 }
#pragma code
#pragma code _LOW_INTERRUPT_VECTOR = 0x000818
void low_ISR (void)
{
;
}
#pragma code
//End bootloader code


#pragma interrupt timer1_isr  //High priority interrupt service routine
 void timer1_isr(void)
 {
   //Pre: The high_ISR function is called.One integer for counting interrupts is defined.
   //Post: LED (from PortB) can shift every second.


  i++;                         //counting the interrupts, this value can increment by one at
every 20 ms.
  PIR1bits.TMR1IF = 0;         //Reset Timer1 interrupt flag
  WriteTimer1(0x8ACF);         //Give new starting value for timer1


   if(i==50)                   // every 1 second the LED can shift once (50*20ms=1 second)
   {
    LATB= LATB<<1;             //shift lights
    if(LATB==0b00000000)
     {
      LATB=0b00000001;         // go back to initial state of LEDs
     }
     i=0;
   }
 }
void main(void)
 {
  //Pre: The timer library is included.
  //Post: The timer1 overflow interrupt can be generated at every 20 ms.

  TRISB = 0x00;                  //Port B output
  LATB=0b00000001;               //initial state of the LEDs.
  ADCON1 = 0b00001111;           //All ADC disabled
  RCONbits.IPEN = 1;             //Enable priority levels on interrupts
  RCONbits.SBOREN = 0;           //Disable BOR

  OpenTimer1( TIMER_INT_ON &     //Interrupt enabled
              T1_8BIT_RW &       //set timer1 as two 8-bit registers
              T1_SOURCE_INT &    //choose Internal clock source (TOSC)
              T1_PS_1_8 &        // Prescale Value: 1:8
              T1_OSC1EN_OFF &    //Disable Timer1 oscillator
              T1_SYNC_EXT_OFF    //Don’t sync external clock input
             );

  WriteTimer1(0x8ACF);           //Set start value of timer,set interrupt at every 20 ms.

  INTCON = 0b10000000;           //enable the high priority interrupts
  IPR1bits.TMR1IP = 1;           //Timer1 interrupt priority high
  PIE1bits.TMR1IE = 1;           //Timer1 interrupt enable



   while(1)
   {
     //add codes here.................

    }
  }
```