```c
#include "Proj_1E_header_file.h"


volatile int m;                                       //Used in ISR but apparently does not need to be volatile?
volatile char overflow;                               //overflow is set to 1 when the most left hand leds are illuminated
unsigned int PORT_1;

int main (void){
unsigned long counter_squared, counter=1;             //32 bits are reserved for each of these variables
m=1;
PORT_1=1;
overflow=0;
setup_HW;
sei();
T1_65ms_clock_tick();                                 //This subroutine starts HW clock Timer 1 that generates an interrupt every 65mS

while(1){                                             //Interrupts enable several process to take place  simultaneously
counter_squared = counter*counter;                    //In this case squares are calculated some of which are printed out
if((!(counter%33))&& (switch_2_down)){                 //"counter%33" is only zero when counter is 33, 66, 99 etc.
Num_to_PC_U(10, counter); Char_to_PC('\t');           //Only print out results if Switch_2 has been pressed
Num_to_PC_U(10, counter*counter); newline();          //Code in this while-loop could be interrupted at any point
 }counter = (counter + 1)%0x10000;                    //limits the value of counter to avoid overflow and garbage out.
Timer_T2_sub(T2_delay_2ms);}
}


ISR(TIMER1_OVF_vect) {
 I2C_Tx_2_integers(PORT_1, ~PORT_1);
 if (m<=5){PORT_1 |= (PORT_1 << 1);m += 1;}           //m += 1; is shorthand for m = m+1;
  else PORT_1 = PORT_1 << 1;                          //once "m" is 6 simply shift the display left
  if(overflow)PORT_1 |= 1;                            //if overflow is 1 execute "PORT_1 |= 1;".
  if (PORT_1 & 0x8000) overflow=1;                    //0x8000 = binary 1000000000000000
  else overflow = 0;
  }
```