```c
#include "Proj_1D_header_file.h"

volatile int p;                                    //p is defined for both main and ISR routines. It is the
                                                   //number of leds that will be skipped every time that
                                                   //the display increments.
int main (void){
long PORT_1=1;
char row=0;                                        //set row to 0 for the top row of leds or 1 for the bottom
setup_HW;
UCSR0B |= (1 << RXCIE0);                           //Enables the serial port Rx interrupt. Sets "RCIE0" without needing to know which bit it is

p=1;                                               //Set "p" to 1 so the the display increments without
sei();                                             //skipping any leds
    while(1){
    if (p <= 8){                                   //If "p" is less than 9
    if (!(row))I2C_Tx_2_integers(PORT_1, 0);       //Illuminate the Upper half of display first and then the lower
    else I2C_Tx_2_integers(0, 0x8000/PORT_1);}     //and  ensure that the lower half is a mirror image of the upper

    else                                           //If "p" is greater than 8 illuminate both upper and lower
    I2C_Tx_2_integers(PORT_1, 0x8000/PORT_1);      //halves of the display together

    Timer_T0_10mS_delay_x_m(12);

    if (p==7)                                      //If p == 7 disable the interrupts and
        {cli();if (waitforkeypress()== 'x')        //make keypresses available to increment display manually
        {sei(); p = 8;}}                           //For a keypress of "x" re-enable interrupts and increment "p"

    PORT_1 = PORT_1 << p;                          //Increment the display by "p"

    if(PORT_1 >= 0x10000){row += 1; row = row%2;   //0x10000 corresponds to leds that do not exist and therefore
    PORT_1 = PORT_1 >> 16;}}}                      //PORT_1 is shifted 16 places to the right to to a led that does exist
                                                   //and the row increments

/*****************Routine executed by a keypress at the PC keyboard if global interrupts are set*******************/

ISR(USART_RX_vect){
switch(receiveChar()){
case '1': if (p==1)p = 15; else p=1; break;        //If user presses key "1": set p to 1 or 15 and exit break block
case '2': if (p==2)p = 14; else p=2; break;        //If user presses key "2": set p to 2 or 14 and exit break block
case '3': if (p==3)p = 13; else p=3; break;        //Continue for keypresses 3 to 7
case '4': if (p==4)p = 12; else p=4; break;        //Other keypresses are ignored
case '5': if (p==5)p = 11; else p=5; break;
case '6': if (p==6)p = 10; else p=6; break;
case '7': p=7; break;}}


/************Local versions of waitforkeypress() and receiveChar()************/

char waitforkeypress_local(void){
while (!(UCSR0A & (1 << RXC0)));     //Bit 7 (RXC0 Receive complete) of UART Control and Status Register A (UCSR0A)
return UDR0;}                        //is set when a character is received by the UART
                                     //Repeat the while-loop endlessly until the bit is set
                                     //then return the contents of UDR0 the I/O register


char receiveChar_local(void)         //We know that a character is already in UDR0 because of the interrupt
{return UDR0;}                       //and immediately return it to the calling routine (i.e. the ISR)

/**********************************************************/
```