src\main.c

```c
// Practice assignment 7, exercise 1

#include <stdio.h>
#include <avr/io.h>
#include <util/delay.h>

#include "i2cmaster.h"
#include "lcd.h" //library init

int main(void) {

  i2c_init(); // initialize I2C and LCD
  LCD_init();

  DDRC = 0xF0; // set data direction for port C pins, 0-3 as input (i.e. the buttons)
  PORTC = 0x3F; // set pull-up resistor for port C
  DDRD = 0xFF; // set data direction for port D, all output
  PORTD= 0x00; // set output for port D (none)

  while(1) { // start program loop

    if(!(PINC & 1)) { // read pin 0 of port C by using 1 (or 00000001) as a mask and then AND'ing it with the pin, giving the last digit in the byte. Then a NOT because the shield uses pull-up resistors, so the default state is 1
      PORTD |= (1 << PIND4); // set pin D4 high by creating mask 00010000 by shifting 1 by 4, then OR'ing it with the PORTD register, thereby setting the pin to high if it isn't already high
    }
    else {
      PORTD &= ~(1 << PIND4); // set pin D4 low by creating mask 11101111 (same as before but with a NOT), then AND'ing, which sets the pin to low if it isn't already low
    }

    if(!((PINC >> 1) & 1)) { //same as before but for pin 1, with mask 00000010, created with the R shift by 1
      PORTD |= (1 << PIND5);
    }
      else {
      PORTD &= ~(1 << PIND5);
    }

    if(!((PINC >> 2) & 1)) { //repeat
      PORTD |= (1 << PIND6);
    }
    else {
      PORTD &= ~(1 << PIND6);
    }

    if(!((PINC >> 3) & 1)) { //repeat
```

```
      PORTD |= (1 << PIND7);
    }
    else {
      PORTD &= ~(1 << PIND7);
    }
    //Setting the LCD after the LED's means multiple LED's can be on at the same time, without the LCD flickering between different messages. Last button gets priority on the LCD

    if(!((PINC >> 3) & 1)) // set LCD, same way to read pin as before
    {
      LCD_set_cursor(0,0);
      printf("DI3 pressed");
    }
    else if(!((PINC >> 2) & 1))
    {
      LCD_set_cursor(0,0);
      printf("DI2 pressed");
    }
    else if(!((PINC >> 1) & 1))
    {
      LCD_set_cursor(0,0);
      printf("DI1 pressed");
    }
    else if(!(PINC & 1))
    {
      LCD_set_cursor(0,0);
      printf("DI0 pressed");
      }
    else // clear LCD by sending it spaces
    {
      LCD_set_cursor(0,0);
      printf("           ");
    }
  }

  return 0;
}

```