```c
/*
 * Objective: implements a timer interrupt handler.
 */

#include <stdio.h>
#include <stdint.h>
#include "eecs388_lib.h"

volatile int intr_count;

#define MAX_INTERRUPTS 16
void (*interrupt_handler[MAX_INTERRUPTS])();
void (*exception_handler[MAX_INTERRUPTS])();

void handle_trap(void) __attribute((interrupt));
void handle_trap()
{
    unsigned long mcause = read_csr(mcause);
    if (mcause & MCAUSE_INT) {
        printf("interrupt. cause=%d, count=%d\n",
            mcause & MCAUSE_CAUSE, (int)intr_count);
        // mask interrupt bit and branch to handler
        interrupt_handler[mcause & MCAUSE_CAUSE] ();
    } else {
        printf("exception=%d\n", mcause & MCAUSE_CAUSE);
        // synchronous exception, branch to handler
        exception_handler[mcause & MCAUSE_CAUSE]();
    }
}

void timer_handler()
{
    intr_count++;

    // YOUR CODE HERE
    //32678
    set_cycles(get_cycles() + (32678 * 0.1));
}

void enable_timer_interrupt()
{
    write_csr(mie, read_csr(mie) | (1 << MIE_MTIE_BIT));
}

void enable_interrupt()
{
    // YOUR CODE HERE
    write_csr(mstatus, read_csr(mstatus) | (1 << MSTATUS_MIE_BIT));
}

void disable_interrupt()
{
    // YOUR CODE HERE
    write_csr(mstatus, read_csr(mstatus) & ~(1 << MSTATUS_MIE_BIT));
}

void register_trap_handler(void *func)
{
    write_csr(mtvec, ((unsigned long)func));
```

```
}

int main (void)
{
    int led_idx = 0;
    int led_gpio[] = {BLUE_LED, GREEN_LED};
    for (int i = 0; i < 2; i++)
        gpio_mode(led_gpio[i], OUTPUT);

    // install timer interrupt handler
    interrupt_handler[7] = timer_handler;

    // write handle_trap address to mtvec
    register_trap_handler(handle_trap);

    // enable timer interrupt
    enable_timer_interrupt();

    // cause timer interrupt
    set_cycles(0);

    // enable global interrupt
    enable_interrupt();

    // main loop.
    int val = 0;
    int prev_intr_count = intr_count;
    while(1) {
        disable_interrupt();
        if (prev_intr_count != intr_count) {
            // toggle led on/off on a new interrupt
            val ^= 1;
            gpio_write(led_gpio[led_idx], val);
            prev_intr_count = intr_count;

            // reset counter at every 10+ interrupts
            if (intr_count >= 10) {
                printf("count=%d. reset\n", (int)intr_count);
                intr_count = 0;
                gpio_write(led_gpio[led_idx], OFF);
                led_idx = (led_idx + 1) % 2;
            }
        }
        enable_interrupt();
    }
    return 0;
}
```