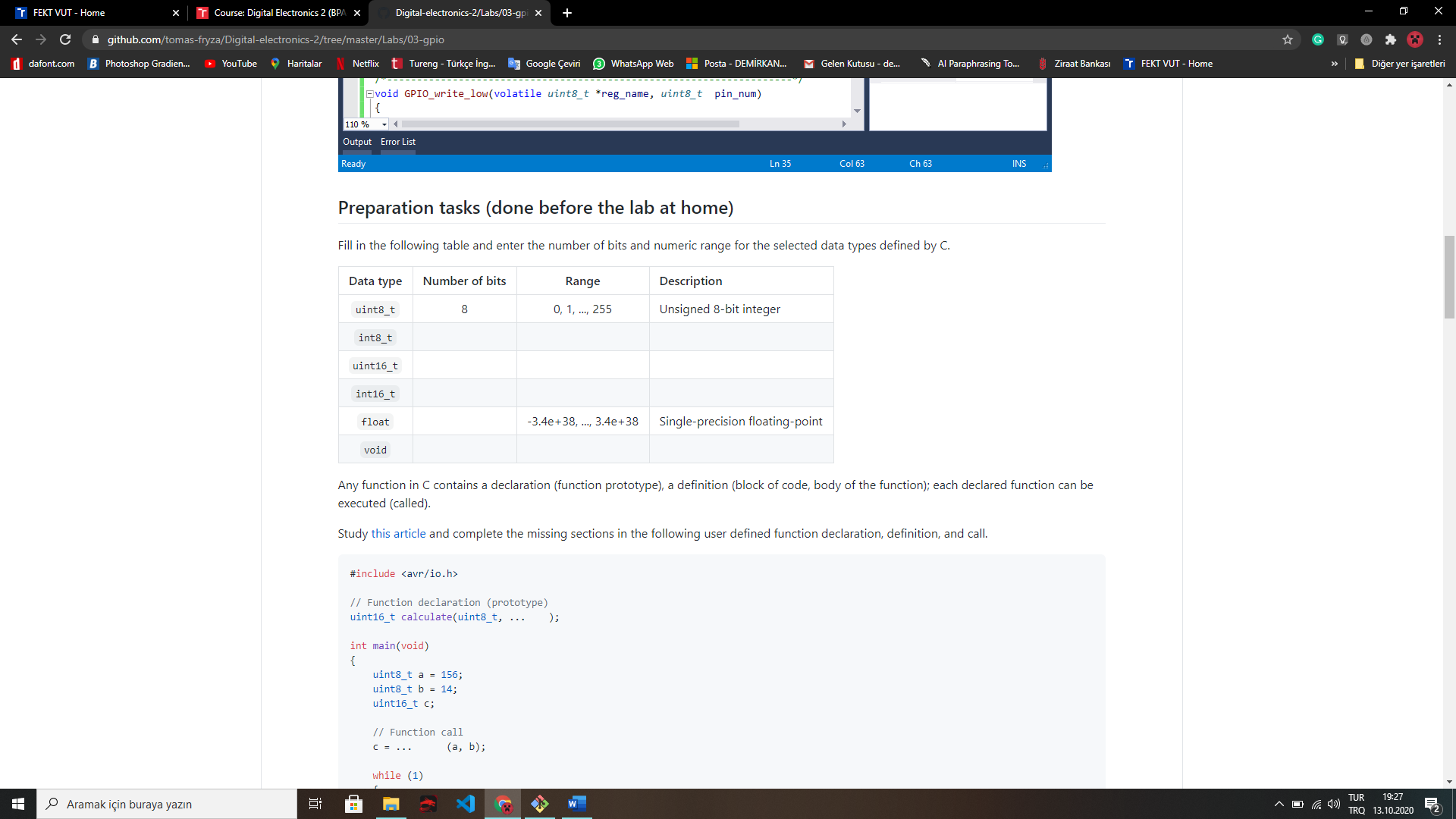
**DIGITAL ELECTRONICS 2 LAB ASSIGNMENT 3**

Name: Demirkan Korbey Baglamac

1. 

0, …, 65 535

16-bit integer

Signed 8-bit integer

Signed 16-bit integer

-32 768, .., +32 767

-128, …, +127

32

16

16

8

-

-

-

Example Code: #include <avr/io.h>

// Function declaration (prototype)

uint16\_t calculate(uint8\_t, uint8\_t);

int main(void)

{

    uint8\_t a = 156;

    uint8\_t b = 14;

    uint16\_t c;

    // Function call

    c = calculate(a, b);

    while (1)

    {

    }

    return 0;

}

// Function definition (body)

uint16\_t calculate(uint8\_t x, uint8\_t y)

{

    uint16\_t result;    // result = x^2 + 2xy + y^2

    result = (x\*x) + (2\*x\*y) + (y\*y);

    return result;

}

1. Listing of library source file gpio.c,

/\* Includes ----------------------------------------------------------\*/

#include "gpio.h"

/\* Function definitions ----------------------------------------------\*/

void GPIO\_config\_output(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name | (1<<pin\_num);

}

/\*--------------------------------------------------------------------\*/

void GPIO\_config\_nopull(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name & ~(1<<pin\_num); // Data Direction Register

\*reg\_name++; // Change pointer to Data Register

\*reg\_name = \*reg\_name & ~(1<<pin\_num); // Data Register

}

/\*--------------------------------------------------------------------\*/

void GPIO\_config\_input\_pullup(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name & ~(1<<pin\_num); // Data Direction Register

\*reg\_name++; // Change pointer to Data Register

\*reg\_name = \*reg\_name | (1<<pin\_num); // Data Register

}

/\*--------------------------------------------------------------------\*/

void GPIO\_write\_low(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name & ~(1<<pin\_num); // Clear bit (and not)

}

/\*--------------------------------------------------------------------\*/

void GPIO\_write\_high(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name | (1<<pin\_num); // Set bit (or)

}

/\*--------------------------------------------------------------------\*/

void GPIO\_toggle(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

\*reg\_name = \*reg\_name ^ (1<<pin\_num); // Toggle bit (xor)

}

/\*--------------------------------------------------------------------\*/

*uint8\_t* GPIO\_read(volatile *uint8\_t* \*reg\_name, *uint8\_t* pin\_num)

{

*uint8\_t* result;

result = \*reg\_name>>pin\_num;

return result;

}

C code of the application main.c,

/\* Defines -----------------------------------------------------------\*/

#define LED\_GREEN PB5 // AVR pin where green LED is connected

#define LED\_RED PC0

#define BTN PD0

#define BLINK\_DELAY 500

#ifndef *F\_CPU*

#define *F\_CPU* 16000000 // CPU frequency in Hz required for delay

#endif

/\* Includes ----------------------------------------------------------\*/

#include <util/delay.h> // Functions for busy-wait delay loops

#include <avr/io.h> // AVR device-specific IO definitions

#include "gpio.h" // GPIO library for AVR-GCC

/\* Function definitions ----------------------------------------------\*/

/\*\*

\* Main function where the program execution begins. Toggle two LEDs

\* when a push button is pressed. Functions from user-defined GPIO

\* library is used instead of low-level logic operations.

\*/

int main(void)

{

/\* GREEN LED \*/

GPIO\_config\_output(&DDRB, LED\_GREEN);

GPIO\_write\_low(&PORTB, LED\_GREEN); //LED off, because active-high

/\* second LED \*/

GPIO\_config\_output(&DDRC, LED\_RED);

GPIO\_write\_low(&PORTC, LED\_RED); //LED off, because active-low

/\* push button \*/

GPIO\_config\_input\_pullup(&DDRD, BTN);

// Infinite loop

while(1)

{

// Pause several milliseconds

*\_delay\_ms*(BLINK\_DELAY);

if(!(GPIO\_read(&PIND, BTN)))

{

GPIO\_toggle(&PORTB, LED\_GREEN);

GPIO\_toggle(&PORTC, LED\_RED);

}

}

// Will never reach this

return 0;

}

In your words, describe the difference between the declaration and the definition of the function in C. Give an example,

The difference between them is function declaration only contains the name of the function and the parameters. But the function definition, as you can understand from the name, contains the body of the function.

For example,

int add(int, int); is a function decleration (prototype). But,

int add(int a, int b) { return a + b; } is a function definition.