Homework

Task 1.

a). DDRB, PORTB, and their combination

DDRB	Description	PORTB	Description
0	Input pin	0	Output low
1	Output pin	1	Output High

DDRB	PORTB	Direction	Internal Pull-up resister	Description
0	0	Input	NO	Tri-state, high-impedance
0	1	Input	YES	Activate pull-up resistor
1	0	Output	NO	Output low (sink)
1	1	Output	NO	Output high (source)

b). Table with inputs and output pins available on Atmega328p

Port	Pin	Input/output usage ?
A	x	Microcontroller ATmega328p does not have port A
B	0	Yes (Arduino pin 8)
	1	Yes (Arduino pin ~9)
	2	Yes (Arduino pin ~10)
	3	Yes (Arduino pin ~11)
	4	Yes (Arduino pin 12)
	5	Yes (Arduino pin 13) - built in LED connected here
	6	N/A
	7	N/A
C	0	Yes (Arduino pin A0)
	1	Yes (Arduino pin A1)
	2	Yes (Arduino pin A2)
	3	Yes (Arduino pin A3)
	4	Yes (Arduino pin A4)
	5	Yes (Arduino pin A5)
	6	N/A
	7	N/A
D	0	Yes (Arduino pin RX<-0)
	1	Yes (Arduino pin TX->1)
	2	Yes (Arduino pin 2)
	3	Yes (Arduino pin ~3)
	4	Yes (Arduino pin 4)
	5	Yes (Arduino pin ~5)
	6	Yes (Arduino pin ~6)
	7	Yes (Arduino pin 7)

c). C code with two LEDS and a push button

```
/*
 * 02-leds.c
 * Created: 9/30/2020 11:39:15
 * Author : masau
 */
/*****************************************************************************
 *
 * Alternately toggle two LEDs when a push button is pressed.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 *****************************************************************************/
/* Defines -----------------------------------------------------------*/
#define LED_GREEN   PB5     // AVR pin where green LED is connected
#define LED_RED     PC0     //AVR pin where red LED is connected
#define BTN         PD0     //AVR pin where the push button is connected
#define BLINK_DELAY 250
#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

/* Functions ---------------------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle two LEDs
 * when a push button is pressed.
 */
int main(void)
{
   /* GREEN LED */
   // Set pin as output in Data Direction Register...
   DDRB = DDRB | (1<<LED_GREEN);
   // ...and turn LED off in Data Register
   PORTB = PORTB & ~(1<<LED_GREEN);

   /* second LED */
   // WRITE YOUR CODE HERE
         DDRC = DDRC | (1<<LED_RED);
         PORTC = PORTC & ~(1<<LED_RED);
         DDRD = DDRD & ~ (1<<BTN);
```

```
        PORTD = PORTD | (1<<BTN);
  // Infinite loop
  while (1)
  {
    // Pause several milliseconds
    _delay_ms(BLINK_DELAY);
                if (bit_is_clear(PIND, BTN))
                {
    // WRITE YOUR CODE HERE
                  PORTB = PORTB ^ (1<<LED_GREEN);
                  PORTC = PORTC ^ (1<<LED_RED);
                }
  }

  // Will never reach this
  return 0;
}
```

Task 2. Knight Rider C code

```
/*
 * Knight_Chaser.c
 *
 * Created: 10/6/2020 14:46:04
 * Author : masau
 */

/*Two different colors of the LED Green and Red are used to observe how the
The toggle is alternating between the LEDS
*/

//Defines
#define LED_GREEN_5   PB5
#define LED_RED_4     PB4
#define LED_GREEN_3   PB3
#define LED_RED_2     PB2
#define LED_GREEN_1   PB1
#define BTN           PD0
#define BLINK_DELAY   50    //
#ifndef F_CPU
#define F_CPU 16000000      // CPU frequency in Hz required for delay
#endif

//Library inclusion
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions

/* Functions ---------------------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle FIVE LEDs
 * when a push button is pressed.
```

```
 */
int main(void)
{
    /* LEDS matching with ports and DDR*/


    DDRB = DDRB | (1<<LED_GREEN_1);  // Set pins as output in Data Direction Register
    PORTB = PORTB & ~(1<<LED_GREEN_1); // ...and turn LED off in Data Register

    DDRB = DDRB | (1<<LED_RED_2); // Set pin as output in Data Direction Register
    PORTB = PORTB & ~(1<<LED_RED_2);  // ...and turn LED off in Data Register

    DDRB = DDRB | (1<<LED_GREEN_3); // Set pin as output in Data Direction Register
    PORTB = PORTB & ~(1<<LED_GREEN_3);

    DDRB = DDRB | (1<<LED_RED_4);     //Set pin as output in Data Direction Register
    PORTB = PORTB & ~(1<<LED_RED_4);  // ...and turn LED off in Data Register

    DDRB = DDRB | (1<<LED_GREEN_5);    //Set pin as output in Data Direction Register
    PORTB = PORTB & ~(1<<LED_GREEN_5);  // ...and turn LED off in Data Register

    DDRD = DDRD & ~(1<<BTN);  //Set pin as input in Data Direction Register
    PORTD = PORTD | (1<<BTN);  //Set pull up resistor on


    // Infinite loop
    while (1)
    {
              // Pause several milliseconds
              _delay_ms(BLINK_DELAY);

                     if(bit_is_clear(PIND,BTN))
                     {

                         //forward toggle
                            PORTB = PORTB ^ (1<<LED_GREEN_1);
                            _delay_ms(BLINK_DELAY);
                            PORTB = PORTB ^ (1<<LED_GREEN_1);
                            PORTB = PORTB ^ (1<<LED_RED_2);
                            _delay_ms(BLINK_DELAY);
                            PORTB = PORTB ^ (1<<LED_RED_2);
                            PORTB = PORTB ^ (1<<LED_GREEN_3);
                            _delay_ms(BLINK_DELAY);
                            PORTB = PORTB ^ (1<<LED_GREEN_3);
                            PORTB = PORTB ^ (1<<LED_RED_4);
                            _delay_ms(BLINK_DELAY);
                            PORTB = PORTB ^ (1<<LED_RED_4);
                            PORTB = PORTB ^ (1<<LED_GREEN_5);
                            _delay_ms(BLINK_DELAY);

                            //backward toggle
                            PORTB = PORTB ^ (1<<LED_GREEN_5);
                            PORTB = PORTB ^ (1<<LED_RED_4);
                            _delay_ms(BLINK_DELAY);
                            PORTB = PORTB ^ (1<<LED_RED_4);
                            PORTB = PORTB ^ (1<<LED_GREEN_3);
                            _delay_ms(BLINK_DELAY);
                            PORTB = PORTB ^ (1<<LED_GREEN_3);
```

```
                        PORTB = PORTB ^ (1<<LED_RED_2);
                        _delay_ms(BLINK_DELAY);
                        PORTB = PORTB ^ (1<<LED_RED_2);
                        PORTB = PORTB ^ (1<<LED_GREEN_1);


                    }
    }
    // Will never reach this
    return 0;
}
```