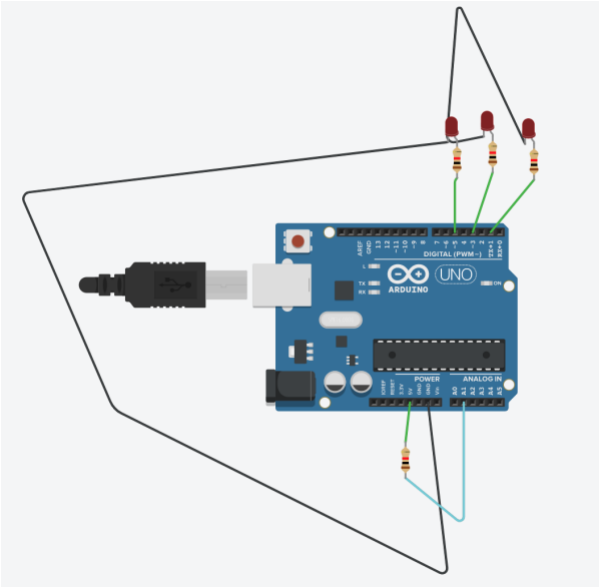
CPE301 – SPRING 2022

Midterm 1

Student Name: David Nakasone
Student #: 2001646072
Student Email: nakasd3@unlv.nevada.edu
Primary Github address: https://github.com/davenakasone
Directory: https://github.com/davenakasone/cpe301_David_Nakasone

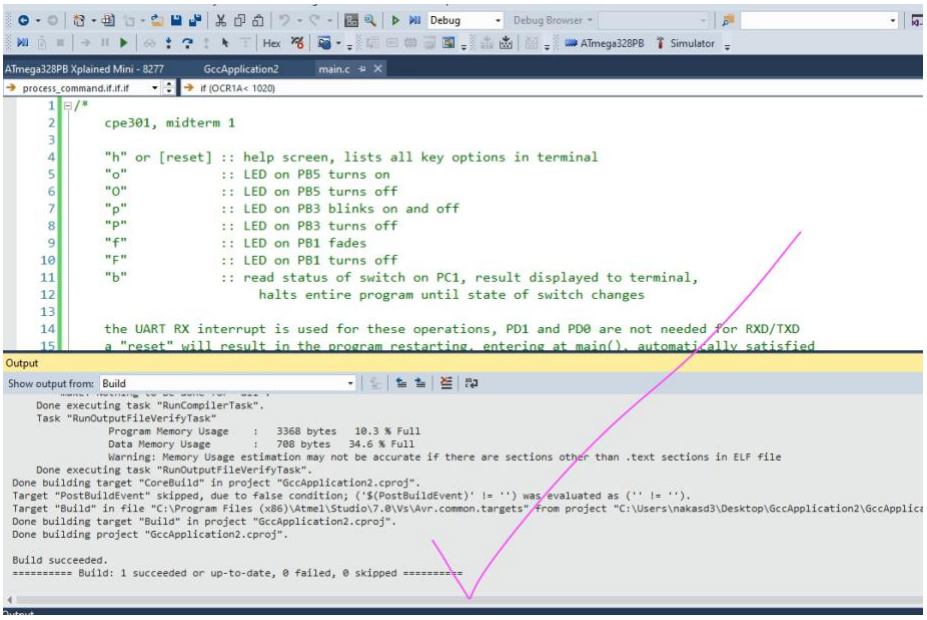
1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

The switch on PC1 is activated by moving the jumper wire to the V_{CC} or ground rail.
The 3 LEDs on PB5, PB3, PB1 are on the bread board.
USART is accessed using the debugWire.



2. INITIAL/MODIFIED/DEVELOPED CODE OF TASKS 1:6

The code builds successfully:



The code for the program, all tasks combined:

```
/*

  cpe301, midterm 1, tasks 1:6


  "h" or [reset] :: help screen, lists all key options in terminal

  "o"          :: LED on PB5 turns on

  "O"          :: LED on PB5 turns off

  "p"          :: LED on PB3 blinks on and off

  "P"          :: LED on PB3 turns off

  "f"          :: LED on PB1 fades

  "F"          :: LED on PB1 turns off

  "b"          :: read status of switch on PC1, result displayed to terminal,

                  halts entire program until state of switch changes


  the UART RX interrupt is used for these operations, PD1 and PD0 are not needed for RXD/TXD

  a "reset" will result in the program restarting, entering at main(), automatically satisfied

  timer3 is set to free run, it will handle PB3 LED by overflow interrupt

  PB1 has OC1A, PWM on timer1 is the best choice to dim the LED


  preparations >>>
```

```
        [hammer] -> toolchain -> AVR/GNU Linker -> General -> check "Use vprintf library(-Wl, -u, vprintf)
        [hammer] -> toolchain -> AVR/GNU Linker -> Miscellaneous -> Other Linker Flags -> "-lprintf_flt"
        tools -> Data Visualizer -> Configuration -> External Connection -> Serial Port ->
            set the terminal's BAUD to 9600, open a terminal, add /r/n, COM3 "mEDBG"
*/

#define F_CPU 16000000UL
#define BAUD 9600
#define BAUD_PRESCALE (((F_CPU / (BAUD * 16UL))) - 1)
#define DATA_BUF 8
#define HELP_BUF 128
#define TRUE 111
#define FALSE 100

#include <avr/interrupt.h>
#include <avr/io.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <util/atomic.h>

const uint16_t RESET_F = 0xAFF;
volatile unsigned char helper [HELP_BUF];
volatile unsigned char data_in[DATA_BUF];
volatile unsigned char command_in;
volatile uint8_t data_count;
volatile uint8_t command_ready;
uint8_t p_active;
uint8_t f_active;
uint8_t f_was_initialized;
uint16_t f_reps;

void usart_putc (char send_char);
void usart_puts (const char* send_str);
void copy_command (void);
void process_command (void);
void help_screen (void);
```

```
int main(void)
{
    // initialize the program variables
    command_ready = FALSE;
    memset(data_in, '\0', DATA_BUF);
    data_count = 0;
    p_active = FALSE;
    f_active = FALSE;
    f_was_initialized = FALSE;
    f_reps = 0;

    DDRB = (1 << 5) | (1 << 3) | (1 << 1);      // PB1, 3, 5 as output
    PORTB = 0;                                  // start with LEDs off
    DDRC = 0;                                   // PC1 as input
    PORTC = (1 << 1);                           // turn on pull-up resistor for PC1

    TIMSK3 = (1 << TOIE3);                      // enable TOV3 overflow interrupt
    TCNT3 = 0;                                  // reset counter
    TCCR3B = (1 << CS32);                       // scale 256, normal mode, timer starts

    UCSR0B |= (1 << RXEN0) | (1 << TXEN0);      // turn on USART module, receive and transmit enabled
    UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);    // configure: asynchronous, 8-bit data, 1-bit stop
    UBRR0H = (BAUD_PRESCALE >> 8);              // sets baud rate
    UBRR0L = BAUD_PRESCALE;
    UCSR0B |= (1 << RXCIE0 );                   // enable RX interrupt
    usart_puts("\r\n");
    usart_puts("        initialization complete, program begins...\r\n");
    usart_puts("\r\n");
    sei();
    help_screen();

    while(1)
    {
        if (command_ready == TRUE)
        {
            copy_command();
```

```
        command_ready = FALSE;
      }
      process_command();
    }
    return EXIT_FAILURE;
}


////~~~~


void copy_command (void)
{
    ATOMIC_BLOCK(ATOMIC_FORCEON)
    {
      command_in = data_in[0];
      memset(data_in, '\0', DATA_BUF);
      usart_puts("\r\n");
      snprintf(helper, HELP_BUF-1,
        "    you entered:  %c\r\n", command_in);
      usart_puts(helper);
      usart_puts("\r\n");
      if (command_in == 'h')
      {
        help_screen();
        return;
      }
      if (command_in == 'b')
      {
        if ((PINC & (1 << 1)))
        {
          usart_puts("the switch was PRESSED, waiting for state change...\n");
          while ((PINC & (1 << 1))) {}
          usart_puts("    switch changed state, exiting...\r\n");
        }
        else
        {
          usart_puts("the switch was NOT PRESSED, waiting for state change...\n");
          while ((PINC & (1 << 1)) == 0) {}
```

```
            usart_puts("    switch changed state, exiting...\r\n");
        }
        return;
    }
    if (command_in != 'o' &&
        command_in != 'O' &&
        command_in != 'p' &&
        command_in != 'P' &&
        command_in != 'f' &&
        command_in != 'F'  )
    {
        snprintf(helper, HELP_BUF-1,
            "[ %c ] is not valid, try again\r\n", command_in);
        usart_puts(helper);
        help_screen();
    }
  }
}


////~~~~


void process_command (void)
{
  f_reps++;
  switch (command_in)
  {
    case ('o') :    // LED on PB5 turns on
        PORTB |= (1 << 5);
        break;
    case ('O') :    // LED on PB5 turns off
        PORTB &= ~(1 << 5);
        break;
    case ('p') :    // LED on PB3 blinks on and off
        p_active = TRUE;
        break;
    case ('P') :    // LED on PB3 turns off
        p_active = FALSE;
```

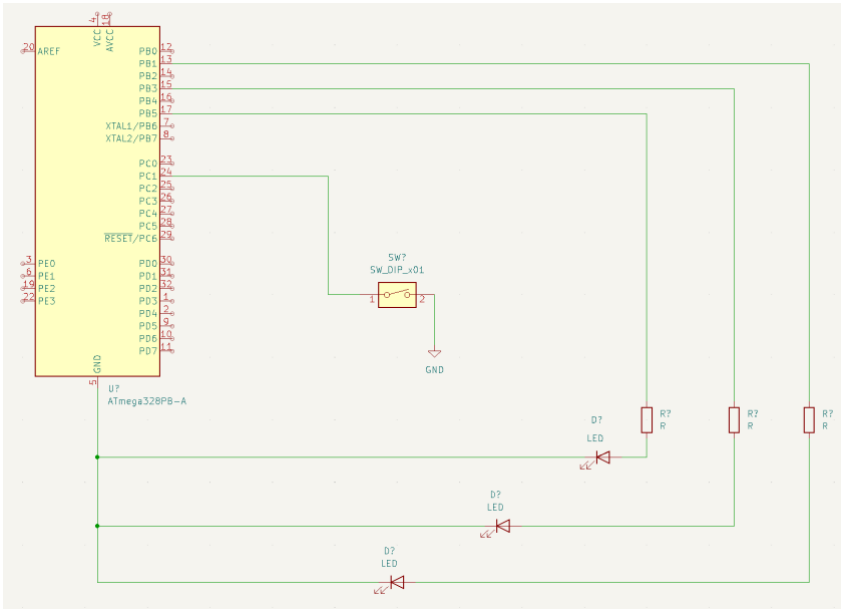
```
        PORTB &= ~(1 << 3);
        break;
    case ('f') :    // LED on PB1 fades
        f_active = TRUE;
        break;
    case ('F') :    // LED on PB1 turns off
        TCCR1B = 0;                 // halt the timer1 PWM
        TCCR1A = 0;
        PORTB &= ~(1 << 1);         // turn of LED and reset trackers
        f_was_initialized = FALSE;
        f_active = FALSE;
        break;
}

if (f_active == TRUE && f_was_initialized == TRUE)
{

    if (f_reps == RESET_F)
    {
        if (OCR1A < 1020)
        {
            OCR1A = OCR1A + 1;
        }
        else
        {
            OCR1A = 1;
        }
        if (OCR1A < 100)
        {
            uint16_t temp = 0;
            while (temp < 0xFFFF)
            {
                temp++;
            }
        }
    }
}
if (f_active == TRUE && f_was_initialized == FALSE)
```

```
  {
    TCNT1 = 0;
    TCCR1C = 0;
    TCCR1A = (1 << COM1A1) | (1 << COM1A0) | (1 << WGM11) | (1 << WGM10);
    TCCR1B = (1 << WGM12) | (1 << CS12);
    OCR1A = 1;
    f_was_initialized = TRUE;
  }
  if (f_reps == RESET_F) {f_reps = 0;}
}


////~~~~


void help_screen (void)
{
  usart_puts("\r\n");
  usart_puts("________HELP MENU >>>\r\n");
  usart_puts("[ h ] or [ reset ]  ,  display this HELP MENU\r\n");
  usart_puts("[ o ] turn on PB5 LED\r\n");
  usart_puts("[ O ] turn off PB5 LED\r\n");
  usart_puts("[ p ] blink PB3 LED\r\n");
  usart_puts("[ P ] turn off PB3 LED\r\n");
  usart_puts("[ f ] fade PB1 LED\r\n");
  usart_puts("[ F ] turn off PB1 LED\r\n");
  usart_puts("[ b ] read status of C1 switch, exits on state change\r\n");
  usart_puts("\r\n");
  usart_puts("________enter your selection:  \r\n");
}


////~~~~


void usart_putc (char send_char)
{
  while ((UCSR0A & (1 << UDRE0)) == 0) {}
  UDR0 = send_char;
```

```
}



////~~~~



void usart_puts (const char* send_str)
{
    while (*send_str)
    {
        usart_putc(*send_str++);
    }
}



////~~~~



ISR (TIMER3_OVF_vect)
{
    if (p_active == TRUE)
    {
        PORTB ^= (1 << 3); // toggle PB3 LED
    }
}



////~~~~



ISR (USART0_RX_vect)
{
    data_in[data_count] = UDR0;
    if (data_in[data_count] == '\n')
    {
        command_ready = TRUE;
        data_count = 0;
    }
    else
    {
```

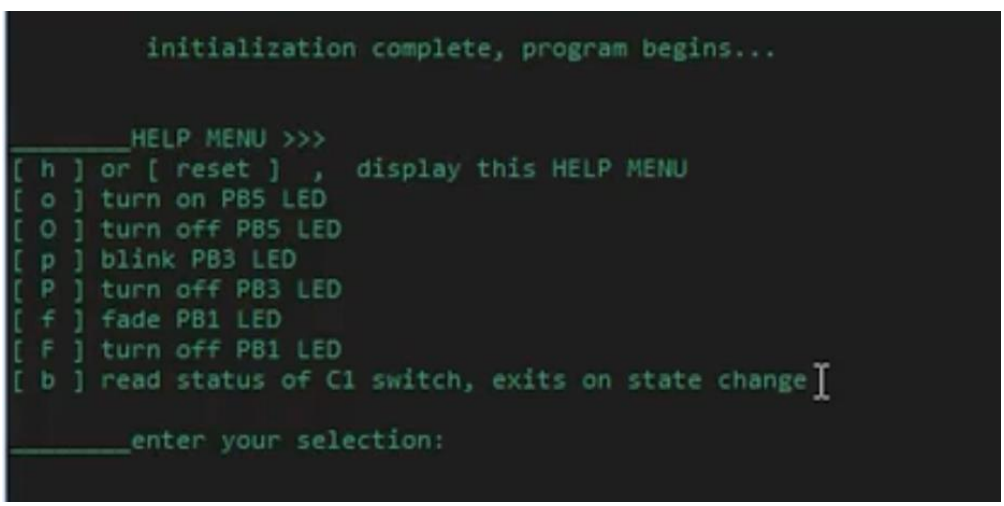
```
        data_count++;
    }
}



////////~~~~~~~~END>  main.c
```

3. SCHEMATICS

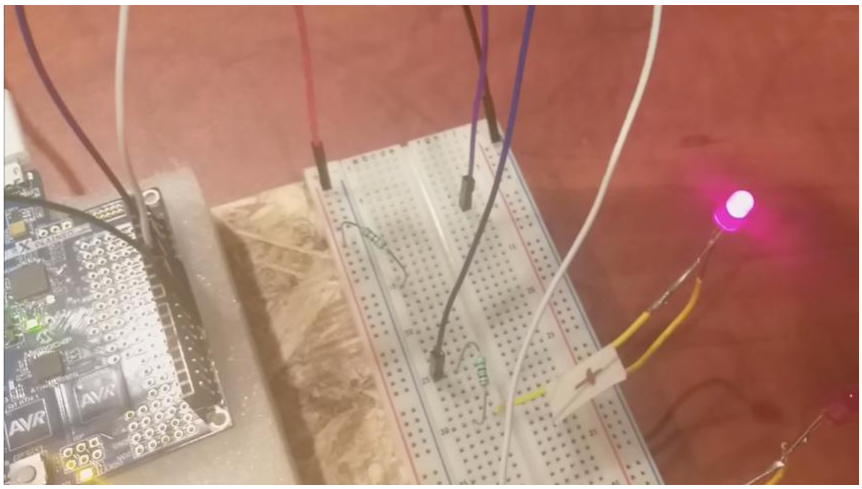


4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

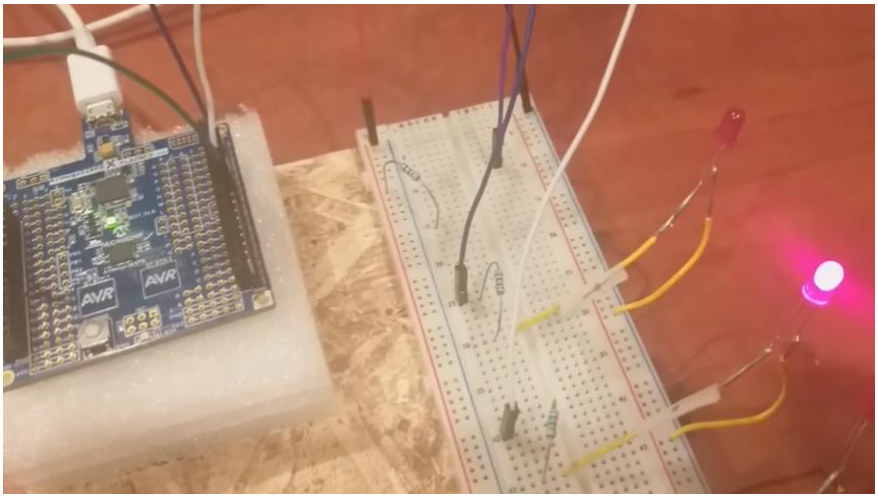
Task1, the help menu:



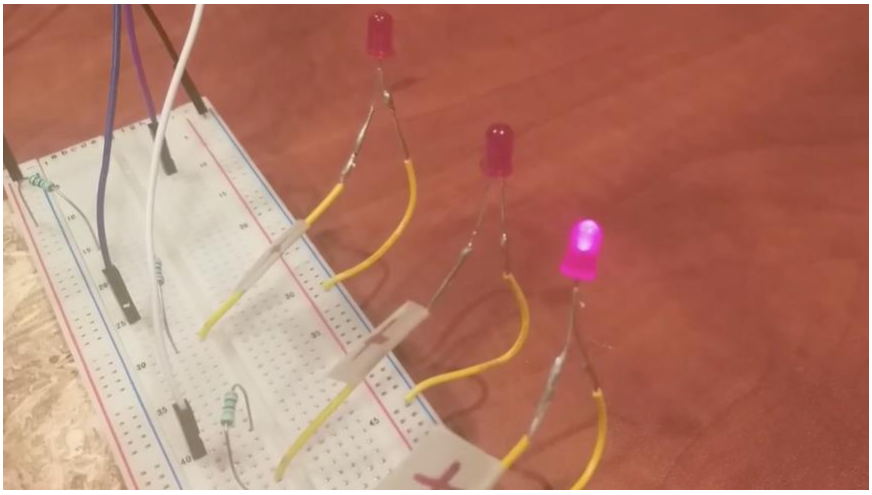
Task 2, turn PB5 LED on:



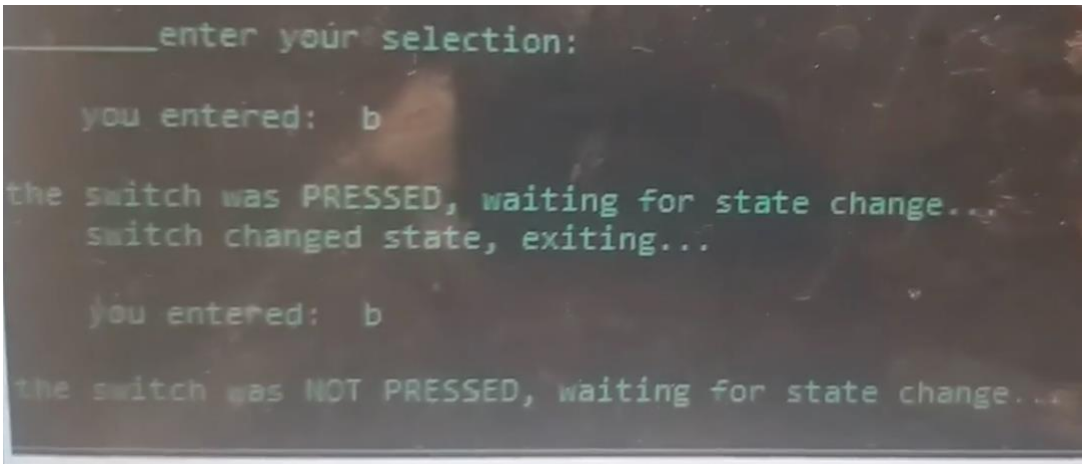
Task 3, blink PB3 LED:



Task 4, fade PB1 LED:

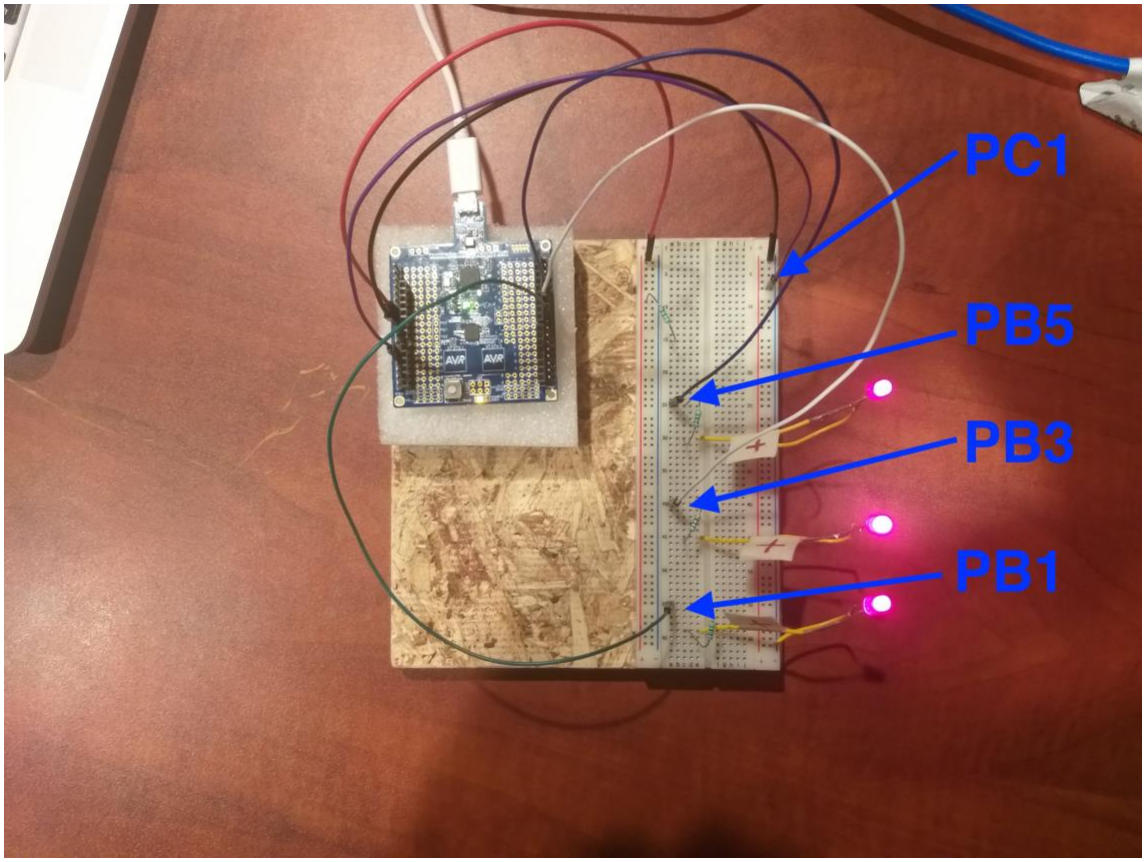


Task 5, manage state change of PC1 switch:



5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Overall board setup, the board setup remains constant between tasks:



6. VIDEO LINKS OF EACH DEMO

task 1: https://youtu.be/-Hw8P-739Lo
task2: https://youtu.be/93O3sy1sWGU
task3: https://youtu.be/eLyFJCjo9OM
task4: https://youtu.be/8nwwCgvvq6c
task5: https://youtu.be/YBeewr9qyNs

7. GITHUB LINK OF THIS DA

https://github.com/davenakasone/cpe301_David_Nakasone/tree/main/Midtermz/Midterm1

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
David Nakasone