1. Write a program to quantize the analog input from a $10k\Omega$ potentiometer fed to channel ADC02, and create a light dimmer that dims and brightens an LED using a PWM signal. Consider the following setup:

    1. A $10k\Omega$ potentiometer is connected to analog input channel 2 (`ADC02`)
    2. The potentiometer is connected to +5V and GND, via suitable resistors.
    3. Utilize the on-board LED (on the Arduino) to create a dimmer routine.

    The potentiometer should be connecteed to an analog input pin. It will put a voltage between 0 and 5 volts on the input pin. This voltage will be converted in the ADC to a digital number between 0 and 1023 acording to the following formula:

    $$ADC_{digital} = \frac{V_{in} * 1023}{V_{ref}} \tag{1}$$

    A lights brightness is can be controlled by a PWM signal. The higher the duty cycle, the brighter the light. The mappings of input voltage, ADC and duty cycle is shown below:

Duty Cycle	**0%**	**25%**	**50%**	**75%**	**100%**
V_{in}	0V	1.25V	2.5V	3.75V	5V
ADC	0	255	511	767	1023

    Since the mapping is linear, you can set up the equation of a line and determine the pulsewidth for the PWM signal for any ADC value between 0-1023.

    Your program should use the free-running mode of the ADC to continuously sample to analog input from the potentiometer. The ADC value should be put into a formula to set the appropriate register to regulate the pulsewidth of the PWM signal.

```
void setup()
{
    DDRC    |= 0x04;
    ADMUX    = 0x43;
    ADCSRA   = 0xAD;
    ADCSRA  |= 0x40;
    TCCR1A   = 0x02;
    TCCR1B   = 0x1B;
    ICR1     = 62535;
    OCR1A    = 0;
}

void loop()
{
    while(1){}
}
ISR(ADC_vect)
{
    byte tempVar = ADCL;
    int pwmChange = 64 * tempVar;
    OCR1A = pwmChange;
}
```

2. A potentiometer can also be used to turn the arm of a servo motor. As the potentiometer goes from 0-5V, the servo motor follows from $-90^\circ \leftrightarrow 90^\circ$. In thatt a servo's angle is controlled by the pulsewidth of the PWM signal, a linear mapping of pulsewidth to ADC value can be defined. The mapping of angle to voltage to ADC value is shown below.

angle ($^\circ$)	**-90**	**-45**	**0**	**45**	**90**
Pulsewidth (ms)	1	1.25	1.5	1.75	2
V_{in} (V)	0	1.25	2.5	3.75	5
ADC	0	255	511	767	1023

Repeat the exercise from problem 1, but this time controlling the servo. Remember that you must come up with an equation of the line so that you can set the pulsewidth for any input voltage on a continuous range.

```
void setup()
{
    DDRC   |= 0x04;
    ADMUX   = 0x43;
    ADCSRA  = 0xAD;
    ADCSRA |= 0x40;
    TCCR1A  = 0x02;
    TCCR1B  = 0x1B;
    ICR1    = 62499;
    OCR1A   = 93;
}

void loop()
{
    while(1){}
}
ISR(ADC_vect)
{
    byte tempVar = ADCL;
    int pwmChange = ((62 / 1023) * tempVar) + 62;
    OCR1A = pwmChange;
}
```