1. Control the movement of a servo motor using the gyroscope from lab. Use the library functions for the MPU6050 used in lab. The servo should be controlled with timer 1, starting at 0 degrees (pulsewidth ), and have a range of motion of

#include <Wire.h>  
#include <MPU6050.h>  
#define SERVOrotate 500  
#define SERVO 375  
#define degree 5  
  
  
MPU6050 mpu; // declare a variable called mpu of datatype MPU6050  
unsigned long timeStampStartOfLoopMs = 0;  
float timeStepS = 0.01;  
float yaw = 0.0f; // pitch, roll and yaw values  
  
unsigned long posistion = 0;  
  
Vector normalizedGyroDPS; //stores the three gyroscope readings XYZ in degrees per second (DPS)  
volatile bool newDataFlag=false; // boolean flag to indicate that timer1 overflow has occurred  
unsigned long startMicroseconds,elapsedMicroseconds; // use for profiling time for certain tasks  
  
  
//==============================================================================  
  
void setup()   
{  
 Serial.begin(115200);  
   
cli();  
TCCR1A = 0b00100010; //Mode 14 fast PWM, 256 scalar  
TCCR1B = 0b00011101;  
TIMSK1 |= 0b00000100; //enable timer 1 overflow interrupt   
ICR1 = 4999;   
OCR1A = SERVO;  
sei();  
  
 while(!mpu.begin(MPU6050\_SCALE\_2000DPS, MPU6050\_RANGE\_2G))   
  
{  
 Serial.println("Could not find a valid MPU6050 sensor, check wiring.");  
 delay(1000);  
}  
  
 mpu.calibrateGyro();   
 mpu.setThreshold(1);  
   
}   
  
//==============================================================================  
  
void loop()  
 {  
 normalizedGyroDPS = mpu.readNormalizeGyro();   
   
 yaw = yaw + normalizedGyroDPS.ZAxis \* timeStepS;  
 Serial.println(yaw);   
  
 posistion = (SERVOrotate - SERVO) \* yaw / degree;  
 OCR1A = SERVO + servoValue;  
}   
  
//==============================================================================

1. Write the instructions necessary to initialize the microcontroller for I2C fast mode to write 0xF0 to the 0xAA register in the peripheral address 0b1001101.

void initI2C (unsigned long bit\_rate){  
 TWBR = ((16000000/bit\_rate)-16)/2; //TWBR set  
 TWCR |= 0b00000100;  
 DDRC &= 0b11001111;   
 PORTC |= 0b00110000; //initialize pins PC4/PC5 as pullups (clocks idle high)  
}  
  
void i2cWaitForComplete(){  
 while(!(TWCR & 0x80)){} //wait until TWINT is true  
}  
  
void i2cStart(){  
 TWCR = 0b10100100; //clear interrupt, initiate start, enable TWI  
 i2cWaitForComplete(); //wait to know start is complete  
}  
  
void i2cStop(){  
 TWCR = 0b10010100; //clear interrupt, initiate stop, enable TWI  
}  
  
void i2cSend(byte data){  
 TWDR = data;  
 TWCR = 0b10000100; //clear interrupt and enable  
 i2cWaitForComplete(); //wait to know data is sent  
}  
  
void setup(){  
 Serial.begin(9600);  
 initI2C(400000);  
}  
  
void loop(){  
 i2cStart();  
 i2cSend(0b10011010);  
 i2cSend(0xAA);  
 i2cSend(0xF0);  
 i2cStop();  
}

1. Write the instructions necessary to initialize the microcontroller for I2C standard mode to read 4 bytes of data from register 0x0F in a peripheral at address 0b1001101. Each byte of data read should be stored in a different variable.

byte read\_data0, read\_data1, read\_data2, read\_data3;  
void initI2C (unsigned long bit\_rate){  
 TWBR = ((16000000/bit\_rate)-16)/2; //TWBR set  
 TWCR |= 0b00000100;  
 DDRC &= 0b11001111;   
 PORTC |= 0b00110000; //initialize pins PC4/PC5 as pullups (clocks idle high)  
}  
  
void i2cWaitForComplete(){  
 while(!(TWCR & 0x80)){} //wait until TWINT is true  
}  
  
void i2cStart(){  
 TWCR = 0b10100100; //clear interrupt, initiate start, enable TWI  
 i2cWaitForComplete(); //wait to know start is complete  
}  
  
void i2cStop(){  
 TWCR = 0b10010100; //clear interrupt, initiate stop, enable TWI  
}  
  
void i2cSend(byte data){  
 TWDR = data;  
 TWCR = 0b10000100; //clear interrupt and enable  
 i2cWaitForComplete(); //wait to know data is sent  
}  
  
byte i2cReadAck(){ //multi-read   
 TWCR = 0b11000100; //clear interrupt, allow ACK, enable TWI  
 i2cWaitForComplete(); //wait to know data is recieved  
 return(TWDR); //return recieved data  
}  
  
byte i2cReadNoAck(){ //single read (always done eventually)  
 TWCR = 0b10000100; //clear interrupt, enable TWI  
 i2cWaitForComplete(); //wait to know data is recieved  
 return(TWDR); //return recieved data  
}  
  
void setup(){  
 Serial.begin(9600);  
 initI2C(100000);  
}  
  
void loop(){  
 i2cStart();  
 i2cSend(0b10011010);  
 i2cSend(0x0F);  
 i2cStart();  
 i2cSend(0b10011011);  
 read\_data0 = i2cReadAck();  
 read\_data1 = i2cReadAck();  
 read\_data2 = i2cReadAck();  
 read\_data3 = i2cReadNoAck();  
 i2cStop();  
}