1. Write a program to interface LEDs with PIC18FXXXX.

```
// Program to interface LEDs with PIC18FXXXX
#include <xc.h>

// Configuration bits (example for PIC18F4550)
#pragma config FOSC = HS        // High-speed oscillator
#pragma config WDT = OFF        // Watchdog Timer off
#pragma config LVP = OFF        // Low-voltage programming off
#pragma config PBADEN = OFF     // PORTB<4:0> as digital I/O

#define _XTAL_FREQ 8000000      // Define oscillator frequency (8 MHz)

void main(void)
{
    TRISB = 0x00;     // Set PORTB as output (LEDs connected to PORTB)
    LATB = 0x00;      // Initialize all LEDs OFF

    while(1)
    {
        LATB = 0xFF;  // Turn ON all LEDs
        __delay_ms(500);

        LATB = 0x00;  // Turn OFF all LEDs
        __delay_ms(500);
    }
}
```

2. Write a program to blink LEDs alternately using PIC18FXXXX.

```
// Program: Blink LEDs alternately using PIC18FXXXX
#include <xc.h>

// Configuration Bits (for PIC18F4550)
#pragma config FOSC = HS        // High-speed crystal oscillator
#pragma config WDT = OFF        // Watchdog Timer disabled
#pragma config LVP = OFF        // Low-voltage programming off
#pragma config PBADEN = OFF     // PORTB<4:0> as digital I/O

#define _XTAL_FREQ 8000000      // 8 MHz oscillator frequency

void main(void)
{
   TRISB = 0x00;   // Set PORTB as output
   LATB = 0x00;    // Initialize LEDs OFF

   while(1)
   {
      LATB = 0xAA;   // 0b10101010 → LEDs ON at even positions
      __delay_ms(500);

      LATB = 0x55;   // 0b01010101 → LEDs ON at odd positions
      __delay_ms(500);
   }
}
```

3.Write a program to interface a button, LED, relay, and buzzer with PIC18FXXXX.

o (a) When Button 1 is pressed, the relay turns ON and LEDs chase in one direction.

o (b) When Button 2 is pressed, the relay turns OFF and LEDs chase in the opposite direction.

```
#include <xc.h>

// Configuration bits for PIC18F4550
#pragma config FOSC = HS        // High-speed oscillator
#pragma config WDT = OFF        // Watchdog Timer disabled
#pragma config LVP = OFF        // Low-voltage programming disabled
#pragma config PBADEN = OFF     // PORTB<4:0> as digital I/O

#define _XTAL_FREQ 8000000      // 8 MHz crystal oscillator

void delay_ms(unsigned int d)
{
   while(d--)
      __delay_ms(10);         // Software delay (adjust as needed)
}

void main(void)
{
   signed char i;
   unsigned char key = 0;

   ADCON1 = 0x0F;      // Configure all pins as digital
   TRISB = 0x00;       // LEDs → Output
   LATB = 0x00;
```

```
// Buttons
TRISAbits.TRISA2 = 1;   // Button 1 input
TRISAbits.TRISA3 = 1;   // Button 2 input

// Relay & Buzzer outputs
TRISAbits.TRISA4 = 0;
TRISAbits.TRISA5 = 0;

LATAbits.LATA4 = 0;     // Relay OFF
LATAbits.LATA5 = 0;     // Buzzer OFF

while(1)
{
   // Read buttons
   if(PORTAbits.RA2 == 0) key = 1;       // Button 1 pressed
   else if(PORTAbits.RA3 == 0) key = 0;  // Button 2 pressed

   if(key == 1)
   {
      LATAbits.LATA4 = 1;   // Relay ON
      LATAbits.LATA5 = 1;   // Buzzer ON

      for(i = 0; i < 8; i++)   // Forward LED chase
      {
         LATB = (1 << i);
         delay_ms(20);
         LATB = 0x00;
         delay_ms(20);
      }
```

```
        }
        else
        {
            LATAbits.LATA4 = 0;   // Relay OFF
            LATAbits.LATA5 = 0;   // Buzzer OFF

            for(i = 7; i >= 0; i--)  // Reverse LED chase
            {
                LATB = (1 << i);
                delay_ms(20);
                LATB = 0x00;
                delay_ms(20);
                if(i == 0) break;   // Prevent underflow
            }
        }
    }
}
```

4. Write a program to interface an LCD with PIC18FXXXX and display your name on the second line.

```
#include <xc.h>

// Configuration bits for PIC18F4550
#pragma config FOSC = HS        // High-speed oscillator
#pragma config WDT = OFF        // Watchdog Timer disabled
#pragma config LVP = OFF        // Low-voltage programming disabled
#pragma config PBADEN = OFF     // PORTB<4:0> as digital I/O

#define _XTAL_FREQ 8000000      // 8 MHz crystal oscillator

// Define LCD pins
#define LCD_RS LATAbits.LATA0
#define LCD_EN LATAbits.LATA1
#define LCDPORT LATB

// Send command to LCD
void lcd_cmd(unsigned char cmd)
{
   LCD_RS = 0;          // Command mode
   LCDPORT = cmd;
   LCD_EN = 1;
   __delay_ms(2);
   LCD_EN = 0;
   __delay_ms(2);
}

// Send data to LCD
void lcd_data(unsigned char data)
{
```

```
    LCD_RS = 1;          // Data mode
    LCDPORT = data;
    LCD_EN = 1;
    __delay_ms(2);
    LCD_EN = 0;
    __delay_ms(2);
}

// Initialize LCD
void lcd_init(void)
{
    lcd_cmd(0x38);   // 8-bit mode, 2-line, 5x7 font
    lcd_cmd(0x0C);   // Display ON, cursor OFF
    lcd_cmd(0x06);   // Increment cursor
    lcd_cmd(0x01);   // Clear display
    __delay_ms(2);
}

void main(void)
{
    ADCON1 = 0x0F;       // All pins as digital
    TRISB = 0x00;        // PORTB as output
    TRISA = 0x00;        // PORTA as output
    LATB = 0x00;
    LATA = 0x00;

    lcd_init();          // Initialize LCD

    lcd_cmd(0xC0);       // Move cursor to 2nd line, 1st position

    // Display name on second line
    lcd_data('H');
```

```
    lcd_data('E');
    lcd_data('M');
    lcd_data('L');
    lcd_data('A');
    lcd_data('T');
    lcd_data('A');

    while(1);            // Hold display
}
```

5. Write a program to interface an LCD with PIC18FXXXX and display your name on the first line.

```
#include <xc.h>

// Configuration bits for PIC18F4550
#pragma config FOSC = HS        // High-speed crystal oscillator
#pragma config WDT = OFF        // Watchdog Timer disabled
#pragma config LVP = OFF        // Low-voltage programming disabled
#pragma config PBADEN = OFF     // PORTB<4:0> as digital I/O

#define _XTAL_FREQ 8000000      // 8 MHz oscillator

// LCD pin definitions
#define LCD_RS LATAbits.LATA0
#define LCD_EN LATAbits.LATA1
#define LCDPORT LATB

// Send command to LCD
void lcd_cmd(unsigned char cmd)
{
   LCD_RS = 0;          // Command mode
   LCDPORT = cmd;
   LCD_EN = 1;
   __delay_ms(2);
   LCD_EN = 0;
   __delay_ms(2);
}

// Send data (character) to LCD
```

```
void lcd_data(unsigned char data)
{
    LCD_RS = 1;          // Data mode
    LCDPORT = data;
    LCD_EN = 1;
    __delay_ms(2);
    LCD_EN = 0;
    __delay_ms(2);
}

// Initialize LCD
void lcd_init(void)
{
    lcd_cmd(0x38);   // 8-bit, 2-line, 5x7 font
    lcd_cmd(0x0C);   // Display ON, cursor OFF
    lcd_cmd(0x06);   // Increment cursor
    lcd_cmd(0x01);   // Clear display
    __delay_ms(2);
}

void main(void)
{
    ADCON1 = 0x0F;       // Configure all pins as digital
    TRISB = 0x00;        // PORTB as output (data)
    TRISA = 0x00;        // PORTA as output (control)
    LATB = 0x00;
    LATA = 0x00;

    lcd_init();          // Initialize LCD
```

```
    lcd_cmd(0x80);        // Set cursor to first line, first position

    // Display your name on the first line
    lcd_data('H');
    lcd_data('E');
    lcd_data('M');
    lcd_data('L');
    lcd_data('A');
    lcd_data('T');
    lcd_data('A');

    while(1);             // Hold display forever
}
```

6. Write a well-documented program to interface a 4×4 keypad with PIC18FXXXX.

```
#include <xc.h>

// CONFIGURATION BITS (adjust as per hardware)
#pragma config FOSC = HS        // High-speed oscillator
#pragma config WDT = OFF        // Watchdog Timer disabled
#pragma config LVP = OFF        // Low-voltage programming disabled
#pragma config PBADEN = OFF     // PORTB<4:0> as digital I/O

#define _XTAL_FREQ 8000000      // 8 MHz crystal frequency

// LCD Pin Definitions
#define LCD_EN  LATAbits.LATA1
#define LCD_RS  LATAbits.LATA0
#define LCDPORT LATB

// Lookup table for 4×4 keypad
const unsigned char KeyLookupTbl[] = {
   '1','4','7','.',
   '2','5','8','0',
   '3','6','9','#',
   'A','B','C','D'
};

// Simple delay for LCD timing
void lcd_delay(unsigned int time)
{
   unsigned int i, j;
```

```
    for(i = 0; i < time; i++)
        for(j = 0; j < 50; j++);
}

// Send command to LCD
void SendInstruction(unsigned char command)
{
    LCD_RS = 0;             // Instruction mode
    LCDPORT = command;
    LCD_EN = 1;
    lcd_delay(5);
    LCD_EN = 0;
    lcd_delay(5);
}

// Send data (character) to LCD
void SendData(unsigned char lcddata)
{
    LCD_RS = 1;             // Data mode
    LCDPORT = lcddata;
    LCD_EN = 1;
    lcd_delay(5);
    LCD_EN = 0;
    lcd_delay(5);
}

// LCD Initialization
void InitLCD(void)
{
    ADCON1 = 0x0F;          // Configure all pins as digital
```

```
    TRISB = 0x00;           // PORTB → output (LCD data)
    TRISAbits.TRISA0 = 0;    // RS pin output
    TRISAbits.TRISA1 = 0;    // EN pin output

    SendInstruction(0x38);   // 8-bit, 2-line, 5x7 display
    SendInstruction(0x0C);   // Display ON, cursor OFF
    SendInstruction(0x06);   // Increment cursor
    SendInstruction(0x01);   // Clear display
    SendInstruction(0x80);   // Set cursor to first line
}

// Function to read a pressed key
unsigned char ReadKey(void)
{
    unsigned char row, val, i, j, key = 0;

    while(1)  // Wait for a key press
    {
        LATD = 0xFF;   // All columns HIGH

        for(i = 0x01; i < 0x10; i = i << 1)  // Scan each column
        {
            LATD = ~i;                    // Drive one column LOW
            lcd_delay(2);
            row = PORTD >> 4;             // Read rows (upper nibble)

            for(j = 0x01; j < 0x10; j = j << 1)  // Check which row is LOW
            {
                if((row & j) == 0)          // Key pressed
                {
```

```
                val = KeyLookupTbl[key];   // Return corresponding ASCII
                return val;
            }
            else
            {
                key++;                  // Increment key index
            }
        }
    }
  }
}

// Main Program
void main(void)
{
    unsigned char Key;
    unsigned char *msg = "Key Pressed = ";

    TRISD = 0xF0;   // RD7–RD4 → Input (Rows), RD3–RD0 → Output (Columns)
    LATD  = 0xFF;   // Initialize PORTD

    InitLCD();      // Initialize LCD

    SendInstruction(0x80);   // Cursor to 1st line
    while(*msg)
        SendData(*msg++);    // Display "Key Pressed ="

    while(1)
    {
        Key = ReadKey();         // Wait for key press
```

```
        SendInstruction(0xC0);   // Move cursor to 2nd line
        SendData(Key);           // Display pressed key
        lcd_delay(100);          // Debounce delay
    }
}
```

7. Generate a square wave using a timer with interrupt (use LED bank).

```
#include <PIC18F4550.h>

// CONFIGURATION BITS
#pragma config FOSC = HSPLL_HS, PLLDIV = 5, CPUDIV = OSC1_PLL2
#pragma config WDT = OFF, LVP = OFF, PBADEN = OFF

unsigned char led_state = 0;

void timer0_init(void)
{
   T0CON = 0b00000111;     // 16-bit mode, prescaler 1:256
   TMR0H = 0xB3;           // Preload for ~1s delay
   TMR0L = 0x0D;
}

void interrupt_init(void)
{
   RCONbits.IPEN = 1;          // Enable priority
   INTCONbits.GIE = 1;         // Enable global interrupts
   INTCONbits.PEIE = 1;        // Enable peripheral interrupts
   INTCONbits.TMR0IE = 1;      // Enable Timer0 interrupt
   INTCONbits.TMR0IF = 0;      // Clear interrupt flag
   INTCON2bits.TMR0IP = 0;     // Low priority
}

void __interrupt(low_priority) LowISR(void)
{
   if (INTCONbits.TMR0IF == 1)
```

```
    {
        TMR0ON = 0;              // Stop timer
        INTCONbits.TMR0IF = 0;   // Clear flag

        // Reload timer for consistent period
        TMR0H = 0xB3;
        TMR0L = 0x0D;

        // Toggle LEDs (square wave output)
        if (led_state == 0)
        {
            LATB = 0xFF;
            led_state = 1;
        }
        else
        {
            LATB = 0x00;
            led_state = 0;
        }

        TMR0ON = 1;              // Restart timer
    }
}

void main(void)
{
    TRISB = 0x00;   // Configure PORTB as output
    LATB = 0x00;    // Start with LEDs off

    interrupt_init();
```

```
    timer0_init();
    TMR0ON = 1;     // Start Timer0

    while (1);      // Infinite loop
}
```

8. Write an embedded C program to display numbers from 0 to 9 on a seven-segment display.

```
#include <pic18f4550.h>

// CONFIGURATION BITS
#pragma config FOSC = HS        // High-speed crystal
#pragma config WDT = OFF        // Watchdog Timer disabled
#pragma config LVP = OFF        // Low-voltage programming disabled

#define _XTAL_FREQ 20000000     // 20 MHz Crystal Frequency

// Lookup table for 0–9 (Common Anode)
const unsigned char segData[10] = {
    0xC0,   // 0
    0xF9,   // 1
    0xA4,   // 2
    0xB0,   // 3
    0x99,   // 4
    0x92,   // 5
    0x82,   // 6
    0xF8,   // 7
    0x80,   // 8
    0x90    // 9
};

// Simple delay
void delay_ms(unsigned int ms)
{
    while(ms--)
```

```
        __delay_ms(1);
}

void main(void)
{
    unsigned char i;

    TRISB = 0x00;   // PORTB → Output (segments)
    LATB  = 0xFF;   // All segments OFF initially

    while(1)
    {
        for(i = 0; i < 10; i++)    // Loop from 0–9
        {
            LATB = segData[i];     // Send pattern to 7-segment
            delay_ms(1000);        // 1 second delay
        }
    }
}
```

9.Write a program to interface a button, LED, relay, and buzzer with PIC18FXXXX.

- **(a) When Button 1 is pressed, the relay turns ON and LEDs chase in one direction.**
- **(b) When Button 2 is pressed, the relay turns OFF and LEDs chase in the opposite direction.**

It is exactly the same as your **PROBLEM STATEMENT 03** in your uploaded file

10. Interface a temperature sensor with Arduino.

```
int sensorPin = A0;    // LM35 output connected to analog pin A0
float temperature;     // Variable to store temperature value

void setup() {
  Serial.begin(9600);  // Initialize Serial Monitor
}

void loop() {
  int sensorValue = analogRead(sensorPin);  // Read analog value (0–1023)

  // Convert ADC value to voltage
  float voltage = sensorValue * (5.0 / 1023.0);

  // LM35 outputs 10mV per °C, so temperature = voltage * 100
  temperature = voltage * 100;

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");

  delay(1000);  // Update every 1 second
}
```

11. Interface a fingerprint sensor with Arduino.

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>

// Define RX and TX pins
SoftwareSerial mySerial(2, 3);  // (RX, TX)
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup() {
  Serial.begin(9600);          // Serial Monitor
  while (!Serial);             // Wait for serial
  delay(100);

  Serial.println("Fingerprint Sensor Test");
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Sensor detected successfully!");
  } else {
    Serial.println("Fingerprint sensor not found :(");
    while (1);  // Halt if not detected
  }

  Serial.println("Place finger on sensor to scan...");
}

void loop() {
  getFingerprintID();   // Continuously check for fingerprint
  delay(1000);
}
```

```
int getFingerprintID() {
  uint8_t p = finger.getImage();

  if (p == FINGERPRINT_NOFINGER) {
    Serial.println("No finger detected.");
    return -1;
  } else if (p != FINGERPRINT_OK) {
    Serial.println("Error capturing image.");
    return -1;
  }

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK) {
    Serial.println("Image conversion failed.");
    return -1;
  }

  p = finger.fingerFastSearch();
  if (p == FINGERPRINT_OK) {
    Serial.print("✅ Match found! ID: ");
    Serial.println(finger.fingerID);
    Serial.print("Confidence: ");
    Serial.println(finger.confidence);
  } else {
    Serial.println("❌ No match found.");
  }
  return finger.fingerID;
}
```

12. Interface a servo motor with Arduino.

```
#include <Servo.h>      // Include Servo library

Servo myservo;          // Create servo object

int pos = 0;            // Variable to store servo position

void setup() {
  myservo.attach(9);    // Attach servo signal pin to D9
  Serial.begin(9600);
  Serial.println("Servo Motor Initialized");
}

void loop() {
  // Rotate servo from 0° to 180°
  for (pos = 0; pos <= 180; pos += 10) {
    myservo.write(pos);           // Move servo to 'pos'
    Serial.print("Angle: ");
    Serial.println(pos);
    delay(500);                   // Wait for servo to reach the position
  }

  // Rotate servo back from 180° to 0°
  for (pos = 180; pos >= 0; pos -= 10) {
    myservo.write(pos);
    Serial.print("Angle: ");
    Serial.println(pos);
    delay(500);
  }
}
```

13. Interface a humidity sensor with Arduino.

```
#include "DHT.h"

#define DHTPIN 7        // Data pin connected to D7
#define DHTTYPE DHT11   // Type of sensor (DHT11 or DHT22)

DHT dht(DHTPIN, DHTTYPE);  // Create DHT object

void setup() {
  Serial.begin(9600);
  Serial.println("DHT11 Humidity and Temperature Sensor");
  dht.begin();  // Initialize sensor
}

void loop() {
  delay(2000);  // Wait 2 seconds between readings

  float humidity = dht.readHumidity();      // Read humidity
  float temperature = dht.readTemperature(); // Read temperature (°C)

  // Check for failed readings
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("⚔ Failed to read from DHT sensor!");
    return;
  }

  Serial.print("🌡 Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
```

```
  Serial.print("💧 Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");
  Serial.println("------------------------");
}
```