1) Project Title: Temperature Monitoring System with AlertsObjective:

Design a system that:

 Monitors the temperature using an LM35 temperature sensor connected to the ADC.

 Displays the temperature on a 16x2 LCD.

 Toggles an LED based on the temperature thresholds using GPIO.

 Responds to a push button interrupt to reset the system.Requirements:

 Hardware:

- ATmega32 Microcontroller
- LM35 Temperature Sensor
- 16x2 LCD Display (connected via 4-bit mode)
- Push Button
- LED (Indicator)

 Functionalities:

- ADC Driver: Convert the analog output from the LM35 to a digital value.
- LCD Driver: Display the temperature in Celsius.
- GPIO Driver: Control the LED based on the temperature threshold.
- Interrupt: Reset the system to default values when the push button is pressed.
- Temperature Thresholds:
    - < 25°C: LED OFF
    - ≥ 25°C: LED ON

□ Bonus Tasks (Optional):

- Add a buzzer for high-temperature warning.
- Use a timer to update the temperature every 1 second.

Project Instructions:

□ ADC Integration:

- Configure the ADC to read the LM35 sensor output (connected to ADC channel 0).
- Convert the ADC value to a temperature using the formula: T(°C)=10mVVout =1024×10mVADC value×Vref
- Use a Vref = 5V.

□ LCD Setup:

- Display the temperature in the format: Temp: XX°C.
- Clear and update the temperature every second.

□ GPIO for LED:

- Configure the LED as output.
- Turn ON the LED if the temperature ≥ 25°C, otherwise turn it OFF.

□ Interrupt for Reset:

- Configure an external interrupt (INT0) on a push button.
- When the button is pressed, reset the temperature display and turn OFF the LED.

□ Program Flow:

- Initialize all drivers: ADC, LCD, GPIO, and Interrupt.
- Continuously read and display the temperature.
- Control the LED based on the temperature.
- Handle button interrupt to reset

2) Project Title: Multi-Sensor Control and Display SystemObjective:

Design a system using ATmega32 microcontroller that:

□ Monitors the temperature (LM35) and light intensity (LDR).

□ Displays the temperature and light intensity percentage on a 16x2 LCD.

□ Uses Push Buttons for mode selection and resetting system values:

- Button 1: Toggle between Celsius (°C) and Fahrenheit (°F).
- Button 2: Toggle display between LCD and Seven Segment Display.
- Button 3: Reset all values and LEDs.

□ Displays temperature or light intensity on a Seven Segment Display in selected mode.

□ Controls two LEDs based on thresholds for temperature and light intensity.System Features:

□ Temperature Thresholds:

- Temperature LED ON: ≥ 30°C.
- OFF otherwise.

□ Light Thresholds:

- Light LED ON: ≥ 70% light intensity.
- OFF otherwise.

□ Modes:

- Default Display Mode: LCD shows both temperature and light intensity.
- Seven Segment Mode: Seven Segment shows either temperature or light intensity based on a toggle.
- Temperature in Seven Segment: Max 99°C (or Fahrenheit equivalent).

Requirements:

 Hardware:

- ATmega32 Microcontroller
- LM35 Temperature Sensor
- LDR (Light Dependent Resistor) with a voltage divider
- 16x2 LCD Display (4-bit mode)
- 3 Push Buttons
- 2 LEDs (Temperature and Light Alerts)
- 1 Common Anode Seven Segment Display
- Resistors for Seven Segment Display

Implementation Details:1. ADC Integration:

- Channel 0: LM35 Sensor for temperature.
- Channel 1: LDR for light intensity.
- Conversion formulas:
    - Temperature (°C): T(°C)=1024ADC value×500
    - Light Intensity (%): Light (%)=(1023ADC value)×100

2. GPIO for LEDs:

- Configure two GPIO pins for controlling Temperature LED and Light LED.
- Turn LEDs ON or OFF based on thresholds.

3. LCD Display:

- Display:
    - Line 1: Temp: XX°C or Temp: XX°F.
    - Line 2: Light: XX%.

4. Seven Segment Display:

- Use PORTC for Seven Segment (common anode).
- Dynamically update the display based on user mode:
    - Temperature or Light Intensity.

5. Push Buttons (Interrupts):

- Button 1 (INT0): Toggle temperature display mode between Celsius and Fahrenheit.
- Button 2 (INT1): Toggle display between LCD and Seven Segment.
- Button 3 (INT2): Reset system to default state.

3) Project: Multi-Mode Seven Segment Controller

Objective:

Design a system using the ATmega32 microcontroller that:

□ Reads input from a potentiometer to set a value between 0 and 99.

□ Displays the value on a two-digit Seven Segment Display.

□ Uses push buttons with interrupts for controlling modes:

- Button 1: Increments the value by 1.
- Button 2: Decrements the value by 1.
- Button 3: Resets the value to 0.

□ Implements an interrupt-driven system for button handling.System Features:

□ Potentiometer Input:

- The potentiometer controls the initial value displayed on the Seven Segment.
- The ADC reads the analog value and maps it to a range from 0 to 99.

□ Seven Segment Display:

- Two digits of a common anode Seven Segment Display are used.
- The value from the potentiometer or push buttons is displayed dynamically.

□ Push Buttons:

- Button 1 (INT0): Increments the displayed value.
- Button 2 (INT1): Decrements the displayed value.
- Button 3 (INT2): Resets the value to 0.

Requirements:

□ Hardware:

- ATmega32 Microcontroller
- Potentiometer
- Common Anode Seven Segment Display (2 digits)
- 3 Push Buttons
- Resistors for Seven Segment Display
- Pull-up resistors for buttons

Implementation Details:1. Potentiometer and ADC:

- ADC Channel 0 reads the potentiometer value.
- Maps the 10-bit ADC value (0–1023) to a 0–99 range: Value=1023ADC Value×99

2. Seven Segment Driver:

- Use PORTC for controlling the Seven Segment Display.
- Dynamically display values for two digits:
    - Multiplex between the tens and units digits using two GPIO pins.

3. Push Button Interrupts:

- INT0 (Button 1): Increment the value (up to 99).
- INT1 (Button 2): Decrement the value (down to 0).
- INT2 (Button 3): Reset the value to 0.

4. Program Flow:

□ Initialize ADC, GPIO, and Interrupts.

□ Read the potentiometer value and map it to 0–99.

□ Display the value on the Seven Segment Display.

□ Handle button presses using interrupts for increment, decrement, and reset.

4)Final Project: Counter with Seven Segment, Push Buttons, and LEDs :-

□ Objective:

Design a system using an ATmega32 microcontroller that:

□ Displays a counter value (0 to 99) on a two-digit Seven Segment Display.

□ Uses push buttons to control the counter:

- Button 1: Increment the counter.
- Button 2: Decrement the counter.
- Button 3: Reset the counter to 0.

□ Uses two LEDs to indicate:

- Green LED: Counter value is even.
- Red LED: Counter value is odd.

System Features:

□ Seven Segment Display:

- Displays a two-digit counter value (0–99).
- Updates dynamically based on button inputs.

□ Push Buttons:

- Button 1: Increment counter (up to 99).
- Button 2: Decrement counter (down to 0).
- Button 3: Reset counter to 0.

□ LED Indicators:

- Green LED turns ON when the counter is even.
- Red LED turns ON when the counter is odd.

Requirements:

□ Hardware:

- ATmega32 Microcontroller
- Common Anode Seven Segment Display (2 digits)
- 3 Push Buttons
- 2 LEDs (Green and Red)
- Resistors for Seven Segment Display and LEDs
- Pull-up resistors for buttons

Implementation Details:1. Seven Segment Driver:

- Use PORTC for controlling the Seven Segment Display.
- Multiplex the two digits:
    - PD0 controls the tens digit.
    - PD1 controls the units digit.

2. Push Button Handling:

- Button 1: Increment the counter (up to 99).
- Button 2: Decrement the counter (down to 0).
- Button 3: Reset the counter to 0.

3. LED Indicators:

- Check if the counter is even or odd:
    - Even: Turn ON the green LED and turn OFF the red LED.
    - Odd: Turn ON the red LED and turn OFF the green LED.

4. Program Flow:

□ Initialize GPIO for LEDs, buttons, and Seven Segment Display.

□ Continuously check button inputs and update the counter.

□ Display the counter value on the Seven Segment Display.

□ Update LED status based on the counter value.