Problem Statement 1: Temperature Monitoring System

Objective: Design a temperature monitoring system that reads temperature data from a sensor and triggers an alarm if the temperature exceeds a predefined threshold.

Requirements:

Read temperature data from a temperature sensor at regular intervals.

Compare the read temperature with a predefined threshold.

If the temperature exceeds the threshold, activate an alarm (e.g., LED or buzzer).

Include functionality to reset the alarm.

Algorithm:

```
1.Initialize temperature sensor and buzzer

2.set a temperature threshold value

3.Read the temperature data from the sensor

4.Compare the read data with threshold value

5.if data is greater the threshold -> Activate alarm

      Otherwise -> ensure alarm is off

6.Reset alarm once it has been triggered
```

Problem Statement 2: Motor Control System

Objective: Implement a motor control system that adjusts the speed of a DC motor based on user input.

Requirements:

Use a potentiometer to read user input for desired motor speed.

Control the motor speed using PWM (Pulse Width Modulation).

Display the current speed on an LCD.

Algorithm:

1.Initialize components potentiometer,motor and LCD

2.Run a loop for continuous evaluation

3.Read the user input value from potentiometer

4.According to the potentiometer value set the motor speed , by adjusting PWM

5.Display each motor speed on LCD

6.repeat the loop

Problem Statement 3: LED Blinking Pattern

Objective: Create an embedded system that controls an array of LEDs to blink in a specific pattern based on user-defined settings.

Requirements:

Allow users to define blink patterns (e.g., fast, slow).

Implement different patterns using timers and interrupts.

Provide feedback through an LCD or serial monitor.

Algorithm:

1.Initialize LED array and buttons(as user interface)

2.Run a loop

3.Check for user input to select or change blink pattern

4.Set selected pattern

5.Implement the selected pattern using timers

6.If fast pattern is selected, Blink LEDs at a high frequency.

7.If slow pattern is selected , Blink LEDs at a lower frequency.

8.If alternate pattern is selected , Blink alternate LEDs in sequence.

9.Repeat the loop for continuous operation.

Problem Statement 4: Data Logger

Objective: Develop a data logger that collects sensor data over time and stores it in non-volatile memory.

Requirements:

Read data from sensors (e.g., temperature, humidity) at specified intervals.

Store collected data in EEPROM or flash memory.

Implement functionality to retrieve and display logged data

Algorithm:

1.Initialize the sensors, memory (EEPROM/flash), and display.

2.Set the time interval for logging data

3.Start a loop to continuously collect and store data

4.Check if the interval has passed:

5.If yes, read data from the sensors

6.Store the data in memory at the next available location.

7.Update the memory location for the next data entry.

8.Repeat the loop to keep logging data at the set interval.

Simple Calculator

Problem Statement: Write a program that functions as a simple calculator. It should be able to perform addition, subtraction, multiplication, and division based on user input.

Requirements:

1. Prompt the user to enter two numbers.

2. Ask the user to select an operation (addition, subtraction, multiplication, division).

3. Perform the selected operation and display the result.

4. Handle division by zero appropriately.

Pseudocode:

```
Enter num1 and num2
Enter the operation = op
if op = +
    then result = num1+ num2
else if op = -
    then result = num1 - num2
else if op = *
    then result = num1 * num2
else if op = /
    then result = num1 / num2
 return the result
```

Factorial Calculation

Problem Statement: Write a program to calculate the factorial of a given non-negative integer.

Requirements:

1. Prompt the user to enter a non-negative integer.

2. Calculate the factorial using a loop.

3. Display the factorial of the number.

Pseudocode (Normal code):

```
Read a non negative number from user = num
if   num is greater than or equal to 0
      factorial =1
       then, Run a loop i=1 to num
             factorial = factorial * i
              return factorial
```

Pseudocode(Using recursion):

```
Read a non negative number from user = num
if   num is greater than or equal to 0, then
function Fact:
           if num less than or equal to 1
                  return 1
            else
                   return num* fact(num-1)
```

Problem Statement: Smart Irrigation System

Objective: Design a smart irrigation system that automatically waters plants based on soil moisture levels and environmental conditions. The system should monitor soil moisture and activate the water pump when the moisture level falls below a predefined threshold.

Requirements:

Inputs:

Outputs:

Conditions:

The pump should only activate if the soil moisture is below the threshold and it is daytime (e.g., between 6 AM and 6 PM).

If the soil moisture is adequate, the system should display a message indicating that watering is not needed.

Activate the water pump when the soil moisture is below the threshold.

Display the current soil moisture level and whether the pump is activated or not.

Soil moisture sensor reading (percentage).

User-defined threshold for soil moisture (percentage).

Time of day (to prevent watering during rain or at night).

Pseudocode:

```
1. Initialize soil moisture threshold(have to be read from user)

2. Set watering StartTime = 6 (6 AM)

3. Set watering EndTime = 18 (6 PM)


4. Function for time check:

    a. Get the current time (hour)

    b. If current time is watering StartTime and watering EndTime ,then

        Return 1 (indicating daytime)

      Else:

        Return 0 (indicating nighttime)


5. Function for calculating soil moisture:

    a. Read soil moisture from sensor (in percentage)

    b. Return the soil moisture


6. Main Program Loop:
```

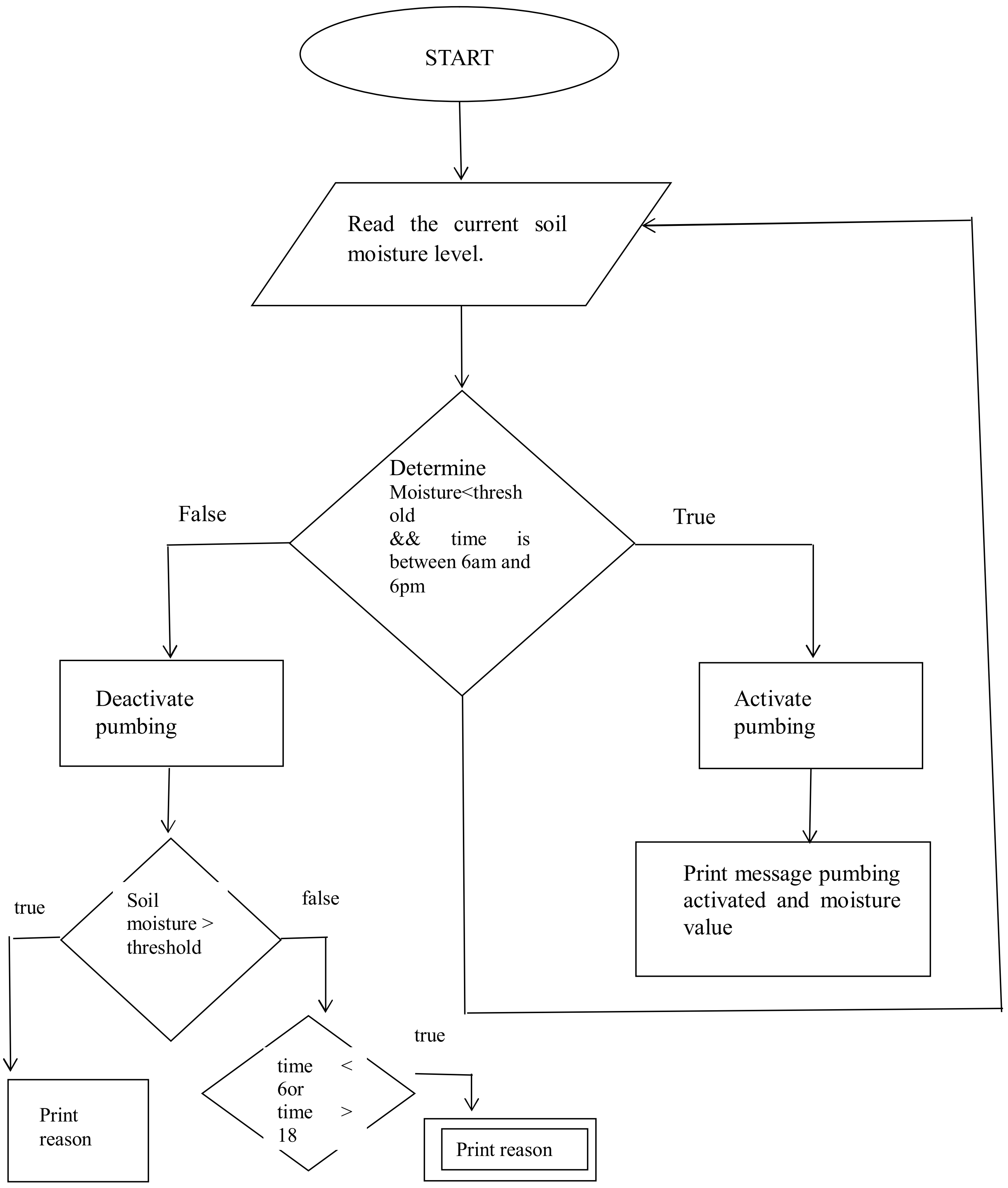
```
a. While system is active:
    i.   currentSoilMoisture = checkSoilMoisture()
    ii.  isDaytime = checkTime()

    iii. If currentSoilMoisture < soilMoistureThreshold AND isDaytime:
             - Activate the water pump
             - Display message "Water pump is activated. " and display Current Soil value in percentage
         Else,
             Deactivate the water pump
             Display the message   "No watering is needed."and display Current Soil Moisture in percentage

    iv. Wait for a specified delay before repeating the loop

7. End Program
```

FLOWCHART:


START
Read the current soil moisture level.
Determine Moisture<threshold && time is between 6am and 6pm
False
True
Deactivate pumbing
Activate pumbing
Print message pumbing activated and moisture value
true
Soil moisture > threshold
false
Print reason
time < 6or time > 18
true
Print reason