SLIP 1

Q1) Java – I/O Decorator (uppercase → lowercase)

```
import java.io.*;

class LowerCaseInputStream extends FilterInputStream {

    protected LowerCaseInputStream(InputStream in) {
        super(in);
    }

    @Override
    public int read() throws IOException {
        int c = super.read();
        return (c == -1) ? c : Character.toLowerCase((char) c);
    }

    @Override
    public int read(byte[] b, int off, int len) throws IOException {
        int n = super.read(b, off, len);
        if (n == -1) return -1;
        for (int i = off; i < off + n; i++) {
            b[i] = (byte) Character.toLowerCase((char) b[i]);
        }
        return n;
    }
}

public class IODecoratorDemo {
    public static void main(String[] args) {
        System.out.println("Enter text: ");
        try (InputStream in = new LowerCaseInputStream(System.in);
```

```
            BufferedReader br = new BufferedReader(new InputStreamReader(in))) {
           String line = br.readLine();
           System.out.println("Lowercase: " + line);
       } catch (IOException e) {
           e.printStackTrace();
       }
   }
}
```

Q2) Arduino C – Sense available Wi-Fi networks (ESP8266)

Connections (Proteus / real board)

- ESP8266 **TX →** Virtual Terminal **RX**
- ESP8266 **RX →** Virtual Terminal **TX**
- ESP8266 **VCC →** 3.3V
- ESP8266 **GND →** GND (also to VT GND)
- Virtual Terminal baud **9600**

```
#include <ESP8266WiFi.h>

void setup() {
  Serial.begin(9600);
  delay(1000);

  Serial.println();
  Serial.println("WiFi Scanner");

  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
}

void loop() {
  Serial.println();
```

```
 Serial.println("Scanning...");

 int n = WiFi.scanNetworks();

 if (n == 0) {
  Serial.println("No networks found");
 } else {
  Serial.print(n);
  Serial.println(" networks found:");
  for (int i = 0; i < n; i++) {
   Serial.print(i + 1);
   Serial.print(". SSID: ");
   Serial.print(WiFi.SSID(i));
   Serial.print("  | RSSI: ");
   Serial.print(WiFi.RSSI(i));
   Serial.println(" dBm");
  }
 }
 delay(10000);
}
```

SLIP 2

Q1) Java – Singleton pattern for multithreading

```
class MySingleton {
    private static volatile MySingleton instance;

    private MySingleton() {}

    public static MySingleton getInstance() {
        if (instance == null) {
            synchronized (MySingleton.class) {
```

```
            if (instance == null) {
                instance = new MySingleton();
            }
        }
    }
    return instance;
  }

  public void showMessage() {
    System.out.println("Singleton instance: " + this);
  }
}

public class SingletonTest {
  public static void main(String[] args) {
    Runnable task = () -> {
      MySingleton s = MySingleton.getInstance();
      s.showMessage();
    };

    new Thread(task).start();
    new Thread(task).start();
    new Thread(task).start();
  }
}
```

Q2) Arduino C – Measure distance & blink LED (Ultrasonic)

Connections

- HC-SR04 **VCC** → 5V
- HC-SR04 **GND** → GND
- HC-SR04 **TRIG** → D9
- HC-SR04 **ECHO** → D8

- LED long leg → 220Ω → D7
- LED short leg → GND

```
const int trigPin = 9;
const int echoPin = 8;
const int ledPin  = 7;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

long getDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  long distance = duration * 0.034 / 2;
  return distance;
}

void loop() {
  long distance = getDistance();
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
```

```
  if (distance < 20) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(500);
}
```

SLIP 3

Q1) Java – Weather station using java.util.Observable

```
import java.util.Observable;
import java.util.Observer;

class WeatherData extends Observable {
    private float temperature, humidity, pressure;

    public void setMeasurements(float t, float h, float p) {
        this.temperature = t;
        this.humidity = h;
        this.pressure = p;
        measurementsChanged();
    }

    public void measurementsChanged() {
        setChanged();
        notifyObservers();
    }

    public float getTemperature() { return temperature; }
    public float getHumidity()    { return humidity; }
    public float getPressure()    { return pressure; }
```

```
}

class CurrentConditionsDisplay implements Observer {
    public CurrentConditionsDisplay(WeatherData wd) {
        wd.addObserver(this);
    }

    @Override
    public void update(Observable o, Object arg) {
        WeatherData wd = (WeatherData) o;
        System.out.println("Temp: " + wd.getTemperature() +
                "  Humidity: " + wd.getHumidity() +
                "  Pressure: " + wd.getPressure());
    }
}

public class WeatherStationDemo {
    public static void main(String[] args) {
        WeatherData wd = new WeatherData();
        new CurrentConditionsDisplay(wd);

        wd.setMeasurements(30f, 70f, 1010f);
        wd.setMeasurements(32f, 65f, 1005f);
    }
}
```

Q2) Arduino C – Detect vibration of an object

Connections

- Vibration sensor SW-420 **VCC →** 5V
- SW-420 **GND →** GND
- SW-420 **DO →** D2
- LED long leg → 220Ω → D13

- LED short leg → GND

```
const int vibrationPin = 2;
const int ledPin = 13;

void setup() {
  pinMode(vibrationPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int val = digitalRead(vibrationPin);

  if (val == HIGH) {
    Serial.println("Vibration detected");
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(200);
}
```

SLIP 4

Q1) Java – Factory Method for Pizza Store

```
abstract class Pizza {
    public void prepare() { System.out.println("Preparing " + getName()); }
    public void bake()    { System.out.println("Baking " + getName()); }
    public void cut()     { System.out.println("Cutting " + getName()); }
    public void box()     { System.out.println("Boxing " + getName()); }
    public abstract String getName();
}
```

```
class NYStyleCheesePizza extends Pizza {
    public String getName() { return "NY Style Cheese Pizza"; }
}

class ChicagoStyleCheesePizza extends Pizza {
    public String getName() { return "Chicago Style Cheese Pizza"; }
}

abstract class PizzaStore {
    public Pizza orderPizza(String type) {
        Pizza pizza = createPizza(type);
        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();
        return pizza;
    }
    protected abstract Pizza createPizza(String type);
}

class NYPizzaStore extends PizzaStore {
    protected Pizza createPizza(String type) {
        if (type.equalsIgnoreCase("cheese")) return new NYStyleCheesePizza();
        return null;
    }
}

class ChicagoPizzaStore extends PizzaStore {
    protected Pizza createPizza(String type) {
        if (type.equalsIgnoreCase("cheese")) return new ChicagoStyleCheesePizza();
```

```
        return null;
    }
}

public class PizzaStoreDemo {
    public static void main(String[] args) {
        PizzaStore ny = new NYPizzaStore();
        PizzaStore ch = new ChicagoPizzaStore();

        ny.orderPizza("cheese");
        ch.orderPizza("cheese");
    }
}
```

Q2) Arduino C – Sense finger on board (Touch sensor)

Connections

- Touch sensor TTP223 **VCC →** 5V
- TTP223 **GND →** GND
- TTP223 **OUT →** D3
- LED long leg → 220Ω → D13
- LED short leg → GND

```
const int touchPin = 3;
const int ledPin   = 13;

void setup() {
  pinMode(touchPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int val = digitalRead(touchPin);
```

```
  if (val == HIGH) {
    Serial.println("Finger sensed");
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(200);
}
```

SLIP 5

Q1) Java – Adapter pattern for Enumeration → Iterator

```
import java.util.Enumeration;
import java.util.Iterator;
import java.util.Vector;

class EnumerationIterator implements Iterator<Object> {
    private Enumeration<?> enumeration;

    public EnumerationIterator(Enumeration<?> e) {
        this.enumeration = e;
    }

    public boolean hasNext() {
        return enumeration.hasMoreElements();
    }

    public Object next() {
        return enumeration.nextElement();
    }
```

```
    public void remove() {
        throw new UnsupportedOperationException();
    }
}

public class EnumerationAdapterDemo {
    public static void main(String[] args) {
        Vector<String> v = new Vector<>();
        v.add("One");
        v.add("Two");
        v.add("Three");

        Enumeration<String> e = v.elements();
        Iterator<Object> it = new EnumerationIterator(e);

        while (it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

Q2) Arduino C – Connect with available Wi-Fi (ESP8266)

Connections

Same as Slip 1 Q2 (ESP8266 + VT).
Baud rate here: **115200** (can also use 9600 if you change code).

```
#include <ESP8266WiFi.h>

const char *ssid     = "Nimisha";     // change to your SSID
const char *password = "nimisha03";   // change to your password

void setup() {
  Serial.begin(115200);
```

```
  delay(1000);

  Serial.println();
  Serial.println("Connecting to WiFi...");

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("Connected to WiFi!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  delay(5000);
}
```

SLIP 6

Q1) Java – Command pattern to test Remote Control

```
interface Command {
    void execute();
}
```

```
class Light {
    public void on()  { System.out.println("Light ON"); }
    public void off() { System.out.println("Light OFF"); }
}

class LightOnCommand implements Command {
    private Light light;
    public LightOnCommand(Light l) { light = l; }
    public void execute() { light.on(); }
}

class LightOffCommand implements Command {
    private Light light;
    public LightOffCommand(Light l) { light = l; }
    public void execute() { light.off(); }
}

class RemoteControl {
    private Command slot;
    public void setCommand(Command c) { slot = c; }
    public void buttonPressed() {
        if (slot != null) slot.execute();
    }
}

public class RemoteControlDemo {
    public static void main(String[] args) {
        Light light = new Light();
        Command on  = new LightOnCommand(light);
        Command off = new LightOffCommand(light);
```

```
        RemoteControl remote = new RemoteControl();
        remote.setCommand(on);
        remote.buttonPressed();
        remote.setCommand(off);
        remote.buttonPressed();
    }
}
```

Q2) Arduino C – Get temperature notification (LM35)

Connections

- LM35 flat side front:
    - Left pin → **5V**
    - Middle pin → **A0**
    - Right pin → **GND**
- LED long leg → 220Ω → D13
- LED short leg → GND

```
const int tempPin = A0;
const int ledPin  = 13;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int adcValue = analogRead(tempPin);
  float voltage = adcValue * (5.0 / 1023.0);
  float tempC   = voltage * 100.0;

  Serial.print("Temp: ");
  Serial.print(tempC);
  Serial.println(" C");
```

```
  if (tempC > 30.0) {
    Serial.println("High Temp Notification!");
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

  delay(1000);
}
```

SLIP 7

Q1) Java – Undo command to test Ceiling Fan

```
interface Command {
    void execute();
    void undo();
}

class CeilingFan {
    public static final int HIGH = 3, MEDIUM = 2, LOW = 1, OFF = 0;
    private int speed = OFF;

    public void high()   { speed = HIGH;   System.out.println("Fan HIGH"); }
    public void medium() { speed = MEDIUM; System.out.println("Fan MEDIUM"); }
    public void low()    { speed = LOW;    System.out.println("Fan LOW"); }
    public void off()    { speed = OFF;    System.out.println("Fan OFF"); }
    public int getSpeed(){ return speed; }
}

class CeilingFanHighCommand implements Command {
  private CeilingFan fan;
```

```
  private int prevSpeed;

  public CeilingFanHighCommand(CeilingFan f) { fan = f; }

  public void execute() {
    prevSpeed = fan.getSpeed();
    fan.high();
  }

  public void undo() {
    switch (prevSpeed) {
      case CeilingFan.HIGH:   fan.high();   break;
      case CeilingFan.MEDIUM: fan.medium(); break;
      case CeilingFan.LOW:    fan.low();    break;
      default:                fan.off();
    }
  }
}

class RemoteWithUndo {
    private Command slot;
    private Command undoCommand;

    public void setCommand(Command c) { slot = c; }
    public void pressButton() {
        slot.execute();
        undoCommand = slot;
    }
    public void pressUndo() {
        if (undoCommand != null) undoCommand.undo();
    }
```

```
}

public class CeilingFanUndoDemo {
    public static void main(String[] args) {
        CeilingFan fan = new CeilingFan();
        Command high = new CeilingFanHighCommand(fan);

        RemoteWithUndo remote = new RemoteWithUndo();
        remote.setCommand(high);
        remote.pressButton();
        remote.pressUndo();
    }
}
```

Q2) Arduino C – LDR to vary light intensity of LED

Connections

- LDR one side → 5V
- Other side of LDR → A0 **and** one end of 10kΩ resistor
- Other end of 10kΩ resistor → GND (voltage divider)
- LED long leg → 220Ω → D9 (PWM)
- LED short leg → GND

```
const int ldrPin = A0;
const int ledPin = 9;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int ldrValue = analogRead(ldrPin);       // 0-1023
  int pwmValue = map(ldrValue, 0, 1023, 0, 255);
```

```
  analogWrite(ledPin, pwmValue);

  Serial.print("LDR: ");
  Serial.print(ldrValue);
  Serial.print("  PWM: ");
  Serial.println(pwmValue);

  delay(200);
}
```

SLIP 8

Q1) Java – State Pattern for Gumball Machine

```
interface State {
    void insertQuarter();
    void ejectQuarter();
    void turnCrank();
    void dispense();
}

class NoQuarterState implements State {
    private GumballMachine machine;
    public NoQuarterState(GumballMachine m) { machine = m; }

    public void insertQuarter() {
        System.out.println("Quarter inserted");
        machine.setState(machine.getHasQuarterState());
    }
    public void ejectQuarter() { System.out.println("No quarter"); }
    public void turnCrank()    { System.out.println("Insert quarter first"); }
    public void dispense()     { System.out.println("Pay first"); }
```

```
}

class HasQuarterState implements State {
    private GumballMachine machine;
    public HasQuarterState(GumballMachine m) { machine = m; }

    public void insertQuarter() { System.out.println("Already have a quarter"); }
    public void ejectQuarter() {
        System.out.println("Quarter returned");
        machine.setState(machine.getNoQuarterState());
    }
    public void turnCrank() {
        System.out.println("Crank turned");
        machine.setState(machine.getSoldState());
    }
    public void dispense() { System.out.println("No gumball dispensed"); }
}

class SoldState implements State {
    private GumballMachine machine;
    public SoldState(GumballMachine m) { machine = m; }

    public void insertQuarter() { System.out.println("Please wait..."); }
    public void ejectQuarter()  { System.out.println("Already turned crank"); }
    public void turnCrank()     { System.out.println("Turning twice doesn't help"); }
    public void dispense() {
        machine.releaseBall();
        if (machine.getCount() > 0)
            machine.setState(machine.getNoQuarterState());
        else {
            System.out.println("Out of gumballs");
```

```
            machine.setState(machine.getSoldOutState());
        }
    }
}

class SoldOutState implements State {
    private GumballMachine machine;
    public SoldOutState(GumballMachine m) { machine = m; }

    public void insertQuarter() { System.out.println("Sold out"); }
    public void ejectQuarter()  { System.out.println("No quarter"); }
    public void turnCrank()     { System.out.println("Sold out"); }
    public void dispense()      { System.out.println("Sold out"); }
}

class GumballMachine {
    private State soldOutState, noQuarterState, hasQuarterState, soldState;
    private State state;
    private int count;

    public GumballMachine(int count) {
        soldOutState  = new SoldOutState(this);
        noQuarterState = new NoQuarterState(this);
        hasQuarterState = new HasQuarterState(this);
        soldState = new SoldState(this);
        this.count = count;
        state = (count > 0) ? noQuarterState : soldOutState;
    }

    public void insertQuarter() { state.insertQuarter(); }
    public void ejectQuarter()  { state.ejectQuarter(); }
```

```
    public void turnCrank() {
        state.turnCrank();
        state.dispense();
    }

    public void releaseBall() {
        if (count > 0) {
            System.out.println("A gumball rolls out...");
            count--;
        }
    }

    public int getCount() { return count; }

    public void setState(State s)         { state = s; }
    public State getSoldOutState()        { return soldOutState; }
    public State getNoQuarterState()      { return noQuarterState; }
    public State getHasQuarterState()     { return hasQuarterState; }
    public State getSoldState()           { return soldState; }
}

public class GumballMachineDemo {
    public static void main(String[] args) {
        GumballMachine gm = new GumballMachine(2);
        gm.insertQuarter();
        gm.turnCrank();
        gm.insertQuarter();
        gm.turnCrank();
        gm.insertQuarter();
    }
}
```

Q2) Raspberry Pi – Execute Linux commands

Write in journal as **steps**, not code:

1. Start Raspberry Pi and login.
2. Open **Terminal**.
3. Run these commands and observe output:

ls

cd /home/pi

touch test.txt

mv test.txt demo.txt

rm demo.txt

mkdir mydir

rmdir mydir

cat /etc/os-release

more /etc/passwd

ps

sudo ls /root

ping google.com

(You can also mention tar, gzip, chown, chgrp, etc.)

SLIP 9

Q1) Java – Simple HR Application using Spring (console style)

(If you don't use actual Spring config in lab, this simple OO version is usually accepted.)

```
public class HRApplication {
    public static void main(String[] args) {
        Employee emp = new Employee(1, "Atharva", "Developer", 60000);
        HRService hr = new HRService();
        hr.printEmployeeDetails(emp);
    }
}


class Employee {
```

```
    private int id;
    private String name;
    private String designation;
    private double salary;

    public Employee(int id, String name, String desig, double sal) {
        this.id = id;
        this.name = name;
        this.designation = desig;
        this.salary = sal;
    }

    @Override
    public String toString() {
        return "Id: " + id + ", Name: " + name +
               ", Designation: " + designation +
               ", Salary: " + salary;
    }
}

class HRService {
    public void printEmployeeDetails(Employee e) {
        System.out.println("Employee Details:");
        System.out.println(e);
    }
}
```

Q2) Python on Pi – Basic programs set 1

```
# a) Read your name and print Hello message
name = input("Enter your name: ")
print("Hello", name)
```

```
# b) Read two numbers and print sum, difference, product, division
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))
print("Sum =", a + b)
print("Difference =", a - b)
print("Product =", a * b)
if b != 0:
    print("Division =", a / b)
else:
    print("Cannot divide by zero")

# c) Word and character count of a given string
s = input("Enter a string: ")
words = s.split()
print("Word count:", len(words))
print("Character count (including spaces):", len(s))

# d) Area of rectangle / triangle / circle
print("1. Rectangle  2. Triangle  3. Circle")
choice = int(input("Enter choice: "))
if choice == 1:
    l = float(input("Length: "))
    br = float(input("Breadth: "))
    print("Area =", l * br)
elif choice == 2:
    b = float(input("Base: "))
    h = float(input("Height: "))
    print("Area =", 0.5 * b * h)
elif choice == 3:
    r = float(input("Radius: "))
    print("Area =", 3.14159 * r * r)
```

```
else:
    print("Invalid choice")
```

SLIP 10

Q1) Java – Strategy Pattern for Duck behavior

```
interface FlyBehavior { void fly(); }
interface QuackBehavior { void quack(); }

class FlyWithWings implements FlyBehavior {
    public void fly() { System.out.println("Flying with wings"); }
}
class FlyNoWay implements FlyBehavior {
    public void fly() { System.out.println("Cannot fly"); }
}

class Quack implements QuackBehavior {
    public void quack() { System.out.println("Quack!"); }
}
class MuteQuack implements QuackBehavior {
    public void quack() { System.out.println("<< Silence >>"); }
}

abstract class Duck {
    protected FlyBehavior flyBehavior;
    protected QuackBehavior quackBehavior;

    public void performFly()  { flyBehavior.fly(); }
    public void performQuack(){ quackBehavior.quack(); }

    public void setFlyBehavior(FlyBehavior fb)   { flyBehavior = fb; }
    public void setQuackBehavior(QuackBehavior qb){ quackBehavior = qb; }
```

```
    public void swim() { System.out.println("Duck swims"); }
    public abstract void display();
}

class MallardDuck extends Duck {
    public MallardDuck() {
        flyBehavior = new FlyWithWings();
        quackBehavior = new Quack();
    }
    public void display() { System.out.println("I'm a Mallard Duck"); }
}

public class DuckStrategyDemo {
    public static void main(String[] args) {
        Duck d = new MallardDuck();
        d.display();
        d.performFly();
        d.performQuack();
        d.setFlyBehavior(new FlyNoWay());
        d.performFly();
    }
}
```

Q2) Python on Pi – set 2 (loops, exception, time, file word count)

```
# a) Print a name 'n' times using for and while
name = input("Enter name: ")
n = int(input("Enter n: "))

print("Using for loop:")
for i in range(n):
    print(name)
```

```
print("Using while loop:")
i = 0
while i < n:
    print(name)
    i += 1

# b) Handle Divide by Zero Exception
try:
    a = int(input("Enter numerator: "))
    b = int(input("Enter denominator: "))
    print("Result:", a / b)
except ZeroDivisionError:
    print("Error: Division by zero")

# c) Print current time 10 times with 10-second interval
import time
from datetime import datetime

for i in range(10):
    now = datetime.now()
    print("Current time:", now.strftime("%H:%M:%S"))
    time.sleep(10)

# d) Read a file line by line and print word count of each line
filename = input("Enter filename: ")
with open(filename, "r") as f:
    for line_no, line in enumerate(f, start=1):
        words = line.split()
        print(f"Line {line_no}: {len(words)} words")
```

SLIP 11

Q1) Java – Adapter pattern: Heart Model to Beat Model

```
interface BeatModel {
   int getBeat();
}

class HeartModel {
   public int getHeartRate() {
      return 72; // dummy value
   }
}

class HeartAdapter implements BeatModel {
   private HeartModel heart;
   public HeartAdapter(HeartModel h) { heart = h; }
   public int getBeat() { return heart.getHeartRate(); }
}

public class HeartAdapterDemo {
   public static void main(String[] args) {
      HeartModel heart = new HeartModel();
      BeatModel beat = new HeartAdapter(heart);
      System.out.println("Beat rate: " + beat.getBeat());
   }
}
```

Q2) Python on Pi – LED, switches, flashing LED

```
# a) Light an LED through Python (GPIO 18)
import RPi.GPIO as GPIO
import time

LED_PIN = 18
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

GPIO.output(LED_PIN, True)
print("LED ON for 5 seconds")
time.sleep(5)
GPIO.output(LED_PIN, False)
GPIO.cleanup()
# b) Two switches, two LEDs
import RPi.GPIO as GPIO
import time

SW1 = 17
SW2 = 27
LED1 = 22
LED2 = 23

GPIO.setmode(GPIO.BCM)
GPIO.setup(SW1, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(SW2, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(LED1, GPIO.OUT)
GPIO.setup(LED2, GPIO.OUT)

try:
    while True:
        GPIO.output(LED1, GPIO.input(SW1) == GPIO.LOW)
        GPIO.output(LED2, GPIO.input(SW2) == GPIO.LOW)
        time.sleep(0.1)
finally:
    GPIO.cleanup()
# c) Flash an LED at on/off times taken from a file
```

```
import RPi.GPIO as GPIO
import time

LED_PIN = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

with open("timing.txt") as f:
    t = f.read().split()
    on_time = float(t[0])
    off_time = float(t[1])

try:
    while True:
        GPIO.output(LED_PIN, True)
        time.sleep(on_time)
        GPIO.output(LED_PIN, False)
        time.sleep(off_time)
finally:
    GPIO.cleanup()
```

SLIP 12

Q1) Java – Decorator pattern for Car → Sports & Luxury

```
interface Car {
    void assemble();
}

class BasicCar implements Car {
    public void assemble() {
        System.out.print("Basic Car");
    }
```

```
}

class CarDecorator implements Car {
    protected Car car;
    public CarDecorator(Car c) { car = c; }
    public void assemble() { car.assemble(); }
}

class SportsCar extends CarDecorator {
    public SportsCar(Car c) { super(c); }
    public void assemble() {
        super.assemble();
        System.out.print(" + Sports features");
    }
}

class LuxuryCar extends CarDecorator {
    public LuxuryCar(Car c) { super(c); }
    public void assemble() {
        super.assemble();
        System.out.print(" + Luxury features");
    }
}

public class CarDecoratorDemo {
    public static void main(String[] args) {
        Car sports = new SportsCar(new BasicCar());
        sports.assemble();
        System.out.println();
        Car luxurySports = new LuxuryCar(new SportsCar(new BasicCar()));
        luxurySports.assemble();
```

```
    }
}
```

Q2) Arduino C – Sense available networks (same as Slip 1 Q2)

Use **exact same ESP8266 Wi-Fi Scanner code and connections** from **Slip 1, Q2**.

SLIP 13

Q1) Java – Adapter design pattern in mobile charger (Volt & Socket)

```
class Volt {
    private int volts;
    public Volt(int v) { volts = v; }
    public int getVolts() { return volts; }
}

class Socket {
    public Volt getVolt() {
        return new Volt(120);
    }
}

interface SocketAdapter {
    Volt get3Volt();
    Volt get12Volt();
    Volt get120Volt();
}

class SocketClassAdapter extends Socket implements SocketAdapter {
    public Volt get3Volt()  { return convertVolt(getVolt(), 40); }
    public Volt get12Volt() { return convertVolt(getVolt(), 10); }
    public Volt get120Volt(){ return getVolt(); }

    private Volt convertVolt(Volt v, int div) {
```

```
        return new Volt(v.getVolts() / div);
    }
}

public class MobileChargerAdapterDemo {
    public static void main(String[] args) {
        SocketAdapter adapter = new SocketClassAdapter();
        System.out.println("3V  : " + adapter.get3Volt().getVolts());
        System.out.println("12V : " + adapter.get12Volt().getVolts());
        System.out.println("120V: " + adapter.get120Volt().getVolts());
    }
}
```

Q2) Arduino C – Ultrasonic + LED (same as Slip 2 Q2)

Reuse **distance + LED** program and connections from **Slip 2, Q2**.

SLIP 14

Q1) Java – Command Design Pattern (Light, GarageDoor, Stereo)

```
interface Command {
    void execute();
}

class Light {
    public void on()  { System.out.println("Light ON"); }
    public void off() { System.out.println("Light OFF"); }
}

class GarageDoor {
    public void up() { System.out.println("Garage Door UP"); }
}

class Stereo {
```

```
    public void onWithCD() { System.out.println("Stereo ON with CD"); }
}

class LightOnCommand implements Command {
    private Light light;
    public LightOnCommand(Light l) { light = l; }
    public void execute() { light.on(); }
}

class LightOffCommand implements Command {
    private Light light;
    public LightOffCommand(Light l) { light = l; }
    public void execute() { light.off(); }
}

class GarageDoorUpCommand implements Command {
    private GarageDoor door;
    public GarageDoorUpCommand(GarageDoor d) { door = d; }
    public void execute() { door.up(); }
}

class StereoOnWithCDCommand implements Command {
    private Stereo stereo;
    public StereoOnWithCDCommand(Stereo s) { stereo = s; }
    public void execute() { stereo.onWithCD(); }
}

class SimpleRemote {
    private Command[] buttons = new Command[4];

    public void setCommand(int slot, Command cmd) {
```

```
        buttons[slot] = cmd;
    }

    public void pressButton(int slot) {
        if (buttons[slot] != null) buttons[slot].execute();
    }
}

public class CommandPatternDemo {
    public static void main(String[] args) {
        Light light = new Light();
        GarageDoor door = new GarageDoor();
        Stereo stereo = new Stereo();

        SimpleRemote remote = new SimpleRemote();
        remote.setCommand(0, new LightOnCommand(light));
        remote.setCommand(1, new LightOffCommand(light));
        remote.setCommand(2, new GarageDoorUpCommand(door));
        remote.setCommand(3, new StereoOnWithCDCommand(stereo));

        remote.pressButton(0);
        remote.pressButton(2);
        remote.pressButton(3);
        remote.pressButton(1);
    }
}
```

Q2) Arduino C – Detect vibration (same as Slip 3 Q2)

Reuse **vibration sensor + LED** program and connections from **Slip 3, Q2**.

SLIP 15

Q1) Java – Facade Pattern for Home Theater

```
class Amplifier {
    public void on()  { System.out.println("Amp ON"); }
    public void off() { System.out.println("Amp OFF"); }
}

class DvdPlayer {
    public void on() { System.out.println("DVD ON"); }
    public void play(String movie) { System.out.println("Playing " + movie); }
    public void off() { System.out.println("DVD OFF"); }
}

class Projector {
    public void on() { System.out.println("Projector ON"); }
    public void wideScreenMode() { System.out.println("Wide Screen Mode"); }
    public void off() { System.out.println("Projector OFF"); }
}

class TheaterLights {
    public void dim(int level) { System.out.println("Lights dim to " + level); }
    public void on() { System.out.println("Lights ON"); }
}

class HomeTheaterFacade {
    private Amplifier amp;
    private DvdPlayer dvd;
    private Projector proj;
    private TheaterLights lights;

    public HomeTheaterFacade(Amplifier a, DvdPlayer d, Projector p, TheaterLights l) {
        amp = a; dvd = d; proj = p; lights = l;
    }
```

```
    public void watchMovie(String movie) {
        System.out.println("Get ready to watch a movie...");
        lights.dim(10);
        proj.on();
        proj.wideScreenMode();
        amp.on();
        dvd.on();
        dvd.play(movie);
    }

    public void endMovie() {
        System.out.println("Shutting movie theater down...");
        lights.on();
        proj.off();
        amp.off();
        dvd.off();
    }
}

public class HomeTheaterDemo {
    public static void main(String[] args) {
        HomeTheaterFacade theater = new HomeTheaterFacade(
                new Amplifier(), new DvdPlayer(), new Projector(), new TheaterLights());

        theater.watchMovie("Avengers");
        theater.endMovie();
    }
}
```

Q2) Arduino C – Sense finger on board (same as Slip 4 Q2)

Reuse **touch sensor + LED** program and connections from **Slip 4, Q2**.

SLIP 16

Q1) Java – Observer Design Pattern for number conversion

```
import java.util.ArrayList;
import java.util.List;

interface ObserverN {
   void update(int number);
}

interface SubjectN {
   void register(ObserverN o);
   void notifyObservers();
}

class NumberSubject implements SubjectN {
   private List<ObserverN> observers = new ArrayList<>();
   private int number;

   public void setNumber(int n) {
      number = n;
      notifyObservers();
   }

   public void register(ObserverN o) {
      observers.add(o);
   }

   public void notifyObservers() {
      for (ObserverN o : observers) {
         o.update(number);
```

```
        }
    }
}

class HexObserver implements ObserverN {
    public void update(int number) {
        System.out.println("Hex: " + Integer.toHexString(number));
    }
}

class OctalObserver implements ObserverN {
    public void update(int number) {
        System.out.println("Octal: " + Integer.toOctalString(number));
    }
}

class BinaryObserver implements ObserverN {
    public void update(int number) {
        System.out.println("Binary: " + Integer.toBinaryString(number));
    }
}

public class NumberObserverDemo {
    public static void main(String[] args) {
        NumberSubject subject = new NumberSubject();
        subject.register(new HexObserver());
        subject.register(new OctalObserver());
        subject.register(new BinaryObserver());

        subject.setNumber(10);
        subject.setNumber(20);
```

```
    }
}
```

Q2) Arduino C – Connect with available Wi-Fi (same as Slip 5 Q2)

Reuse **Wi-Fi connect** code and ESP8266 connections from **Slip 5, Q2**.

SLIP 17

Q1) Java – Abstract Factory Pattern for Shape

```
interface Shape {
    void draw();
}

class Rectangle implements Shape {
    public void draw() { System.out.println("Drawing Rectangle"); }
}

class Circle implements Shape {
    public void draw() { System.out.println("Drawing Circle"); }
}

interface ShapeFactory {
    Shape getShape(String type);
}

class NormalShapeFactory implements ShapeFactory {
    public Shape getShape(String type) {
        if (type.equalsIgnoreCase("rectangle")) return new Rectangle();
        return null;
    }
}

class RoundedShapeFactory implements ShapeFactory {
```

```
    public Shape getShape(String type) {
        if (type.equalsIgnoreCase("circle")) return new Circle();
        return null;
    }
}

class FactoryProducer {
    public static ShapeFactory getFactory(String choice) {
        if (choice.equalsIgnoreCase("rounded")) return new RoundedShapeFactory();
        return new NormalShapeFactory();
    }
}

public class AbstractFactoryDemo {
    public static void main(String[] args) {
        ShapeFactory normal = FactoryProducer.getFactory("normal");
        Shape rect = normal.getShape("rectangle");
        rect.draw();

        ShapeFactory rounded = FactoryProducer.getFactory("rounded");
        Shape circle = rounded.getShape("circle");
        circle.draw();
    }
}
```

Q2) Arduino C – Temperature notification (same as Slip 6 Q2)

Reuse **LM35 + LED** code and wiring from **Slip 6, Q2**.

SLIP 18

Q1) Java – Weather Station using Observable (same as Slip 3 Q1)

Reuse **WeatherStationDemo** code from **Slip 3, Q1**.

Q2) Arduino C – LDR vary light intensity (same as Slip 7 Q2)

Reuse **LDR + PWM LED** code and wiring from **Slip 7, Q2**.