Slip 1

Slip 1a

Q1 Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters. [15 M]

```
/*
Slip 1a
Q1 Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters. [15 M]
*/

import java.io.*;

public class slip1a {
    public static void main(String[] args) {
        try {
            // Read input from terminal
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Enter text in UPPERCASE: ");
            String inputText = br.readLine();

            // Convert each character to lowercase
            StringWriter sw = new StringWriter();
            BufferedWriter bw = new BufferedWriter(sw);

            for (int i = 0; i < inputText.length(); i++) {
                char lowerChar = Character.toLowerCase(inputText.charAt(i));
                bw.write(lowerChar);
            }

            bw.close();

            // Display converted text
            System.out.println("Converted Text: " + sw.toString());

        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

/*
-----------------------------------------------
```

```
 TERMINAL COMMANDS (VS CODE)
------------------------------------------------
javac slip1a.java
java slip1a

------------------------------------------------
 SAMPLE INPUT (typed in terminal)
------------------------------------------------
HELLO JAVA DECORATOR

------------------------------------------------
 SAMPLE OUTPUT
------------------------------------------------
Converted Text: hello java decorator
------------------------------------------------
*/
```

Slip 1b

Q2 Write a program to sense the available networks using Arduino

```
#include <SPI.h>
#include <WiFi.h>
void setup() {
// initialize serial and wait for the port to open: Serial.begin(9600); while
(Serial)
:
// attempt to connect using WEP
encryption:
Serial.println("Initializing Wifi...");
printMacAddress();
// scan for existing networks:
Serial.println("Scanning available
networks..."); listNetworks(); } void loop() {
delay(1000
0);
// scan for existing networks:
Serial.println("Scanning available networks..."); listNetworks();
```

```
Serial.println("Scanning available
networks..."); listNetworks();
}
void printMacAddress() {
// the MAC address of your Wifi shield byte
mac [6];
// print your MAC address:
WiFi.macAddress(mac);
Serial.print("MAC: ");
Serial.print(mac[5], HEX);
Serial.print(":");
Serial.print(mac [4], HEX);
Serial.print(":");
Serial.print(mac [3], HEX);
Serial.print(":");
Serial.print(mac [2], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.println(mac[0], HEX);
} void listNetworks()
{
// scan for nearby networks:
Serial.println("** Scan Networks **");
byte numSsid
WiFi.scanNetworks();
// print the list of networks seen:
Serial.print("number of available
networks:");
Serial.println(numSsid);
// print the network number and name
for each network found:
for (int thisNet = 0; thisNet
```

```
< numSsid; thisNet++) {
Serial.print(thisNet);
Serial.print(") ");
Serial.print(WiFi.SSID (thisNet));
Serial.print("\tSignal: ");
Serial.print(WiFi.RSSI(thisNet));
Serial.print(" dBm);
Serial.print("\tEncryption: ");
Serial.println(WiFi.encryptionType
(thisNet));
}
}
```

Slip 2
Slip 2a
Q1 Write a Java Program to implement Singleton pattern for multithreading

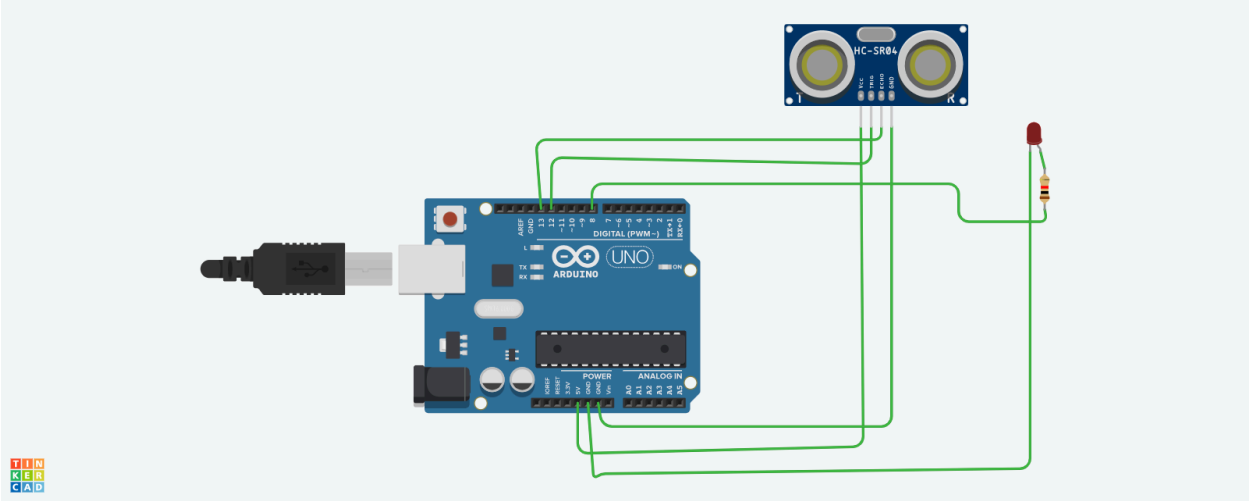
```
/*
Slip 2a
Q1 Write a Java Program to implement Singleton pattern for multithreading
*/

class Singleton {
    // volatile ensures visibility across threads
    private static volatile Singleton instance;

    // private constructor to prevent external instantiation
    private Singleton() {}

    // thread-safe method to get the singleton instance
    public static Singleton getInstance() {
        if (instance == null) { // first check
            synchronized (Singleton.class) {
                if (instance == null) { // double-check
                    instance = new Singleton();
                }
            }
        }
        return instance;
    }

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

```
public class slip2a {
    public static void main(String[] args) {
        // Create multiple threads to test singleton
        Runnable task = () -> {
            Singleton obj = Singleton.getInstance();
            obj.showMessage();
        };

        Thread t1 = new Thread(task);
        Thread t2 = new Thread(task);
        Thread t3 = new Thread(task);

        t1.start();
        t2.start();
        t3.start();
    }
}
/*
-----------------------------------------------
 TERMINAL COMMANDS (VS CODE)
-----------------------------------------------
javac slip2a.java
java slip2a

-----------------------------------------------
 SAMPLE OUTPUT
-----------------------------------------------
Singleton instance: Singleton@1b6d3586
Singleton instance: Singleton@1b6d3586
Singleton instance: Singleton@1b6d3586
(All threads share the same singleton instance)
-----------------------------------------------
*/
```

Slip 2b
Q2 Write a program to measure the distance using ultrasonic sensor and make LED blink using Arduino.

```
// Pin definitions
const int trig = 12;
const int echo = 13;
const int LED1 = 8;

int duration = 0;
int distance = 0;

void setup() {
  pinMode(trig, OUTPUT);   // Trigger pin sends ultrasonic pulse
  pinMode(echo, INPUT);    // Echo pin receives reflected pulse
  pinMode(LED1, OUTPUT);   // LED output

  Serial.begin(9600);      // Start serial monitor
}

void loop() {

  // Send a HIGH pulse to trigger ultrasonic
  digitalWrite(trig, HIGH);
  delayMicroseconds(1000);   // Important: pulse duration for sensor
  digitalWrite(trig, LOW);

  // Read how long echo pin stayed HIGH (time of reflection)
  duration = pulseIn(echo, HIGH);

  // Convert time into distance in cm
```

```
  distance = (duration / 2) / 28.5;   // Speed of sound based formula

  Serial.println(distance);   // Print distance on Serial Monitor

  // Check if object is close (≤ 7 cm)
  if (distance <= 7) {
    digitalWrite(LED1, HIGH);   // LED ON when object is close
  }
  else {
    digitalWrite(LED1, LOW);    // LED OFF when object is far
  }
}
```



Arduino UNO,
Ultrasonic,
Register,
LED

Slip 3
Slip 3a
Q1 Write a JAVA Program to implement built-in support (java.util.Observable) Weather station with members temperature, humidity, pressure and methods mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(), getPressure()

```
/*
Slip 3a
Q1 Write a JAVA Program to implement built-in support (java.util.Observable)
Weather station
with members temperature, humidity, pressure and methods
measurementsChanged(),
setMeasurements(), getTemperature(), getHumidity(), getPressure()
*/

// Custom observer pattern (modern Java, no deprecated Observable)
interface Observer {
    void update(float temp, float hum, float pres);
}

class Weather {
    private float temp;
    private float hum;
    private float pres;
    private java.util.List<Observer> observers = new java.util.ArrayList<>();

    public void addObserver(Observer o) {
        observers.add(o);
    }
```

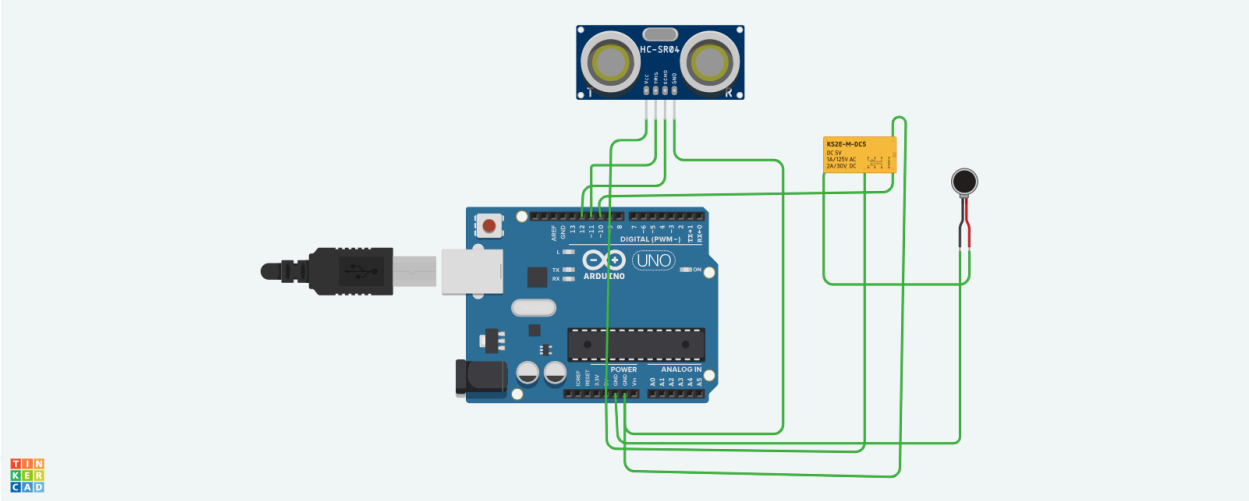
```
    public void setMeasurements(float temp, float hum, float pres) {
        this.temp = temp;
        this.hum = hum;
        this.pres = pres;
        measurementsChanged();
    }

    private void measurementsChanged() {
        for (Observer o : observers) {
            o.update(temp, hum, pres);
        }
    }

    public float getTemperature() { return temp; }
    public float getHumidity() { return hum; }
    public float getPressure() { return pres; }
}

class Display implements Observer {
    private String name;
    public Display(String name) { this.name = name; }

    public void update(float temp, float hum, float pres) {
        System.out.println(name + " -> Temp: " + temp + ", Hum: " + hum + ",
Pres: " + pres);
    }
}

public class slip3a {
    public static void main(String[] args) {
        Weather w = new Weather();

        Display d1 = new Display("Main");
```

```
        Display d2 = new Display("Side");

        w.addObserver(d1);
        w.addObserver(d2);

        // Simulate measurement changes
        w.setMeasurements(30.5f, 65.0f, 1013.0f);
        w.setMeasurements(32.0f, 70.0f, 1012.5f);
    }
}

/*
-----------------------------------------------
 TERMINAL COMMANDS (VS CODE)
-----------------------------------------------
javac slip3a.java
java slip3a

-----------------------------------------------
 SAMPLE OUTPUT
-----------------------------------------------
Main -> Temp: 30.5, Hum: 65.0, Pres: 1013.0
Side -> Temp: 30.5, Hum: 65.0, Pres: 1013.0
Main -> Temp: 32.0, Hum: 70.0, Pres: 1012.5
Side -> Temp: 32.0, Hum: 70.0, Pres: 1012.5
-----------------------------------------------
*/
```

Slip 3b
Q2 Write a program to detects the vibration of an object with sensor using Arduino.

```
// Pin definitions
const int UltrasonikTrig1 = 11;
const int UltrasonikEcho1 = 12;
const int Indikator = 10;

// Variables
long duration1;
long cm1;

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

  pinMode(UltrasonikTrig1, OUTPUT);   // Trigger pin for ultrasonic
  pinMode(UltrasonikEcho1, INPUT);    // Echo pin for ultrasonic
  pinMode(Indikator, OUTPUT);         // LED or indicator output
}

void loop() {
  // Send pulse to ultrasonic sensor
  digitalWrite(UltrasonikTrig1, LOW);
  delayMicroseconds(2);

  digitalWrite(UltrasonikTrig1, HIGH);
  delayMicroseconds(10);
  digitalWrite(UltrasonikTrig1, LOW);

  // Read the echo return time
  duration1 = pulseIn(UltrasonikEcho1, HIGH);
```

```
  // Convert time to centimeters
  cm1 = microsecondsKeCenti(duration1);

  // Print distance value
  Serial.print("Sensor 1: ");
  Serial.print(cm1);
  Serial.print(" cm  ||  ");

  // If distance is less than or equal to 100 cm
  if (cm1 <= 100) {
    digitalWrite(Indikator, HIGH);   // Object close → Indicator ON
  } else {
    digitalWrite(Indikator, LOW);    // Object far → Indicator OFF
    Serial.print(" Safe");
  }

  Serial.println(" ||");
  delay(100);
}

// Convert microseconds to centimeters
long microsecondsKeCenti(long microseconds) {
  return microseconds / 29 / 2;
}
```



ultrasonic distance sensor
vibration motor
arduino uno r3
Relay DPDT

Slip 4
Slip 4a
Q1 Write a Java Program to implement Factory method for Pizza Store with createPizza(), orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizza's like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.
/*
Slip 4a
Q1 Write a Java Program to implement Factory method for Pizza Store with createPizza(),
orderPizza(), prepare(), bake(), cut(), box(). Use this to create variety of pizzas like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.
*/

```
// Abstract Pizza class
abstract class Pizza {
    String name;

    public void prepare() {
        System.out.println("Preparing " + name);
    }

    public void bake() {
        System.out.println("Baking " + name);
    }

    public void cut() {
        System.out.println("Cutting " + name);
    }

    public void box() {
        System.out.println("Boxing " + name);
    }
```

```
    public String getName() { return name; }
}

// Concrete Pizzas
class NYStyleCheesePizza extends Pizza {
    public NYStyleCheesePizza() { name = "NY Style Cheese Pizza"; }
}

class ChicagoStyleCheesePizza extends Pizza {
    public ChicagoStyleCheesePizza() { name = "Chicago Style Cheese Pizza"; }
}

// Pizza Store abstract class
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);
}

// Concrete Pizza Stores
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 slip4a {
    public static void main(String[] args) {
        PizzaStore nyStore = new NYPizzaStore();
        PizzaStore chicagoStore = new ChicagoPizzaStore();

        Pizza pizza1 = nyStore.orderPizza("cheese");
        System.out.println("Ordered: " + pizza1.getName());

        Pizza pizza2 = chicagoStore.orderPizza("cheese");
        System.out.println("Ordered: " + pizza2.getName());
    }
}

/*
-----------------------------------------------
 TERMINAL COMMANDS (VS CODE)
-----------------------------------------------
javac slip4a.java
java slip4a

-----------------------------------------------
 SAMPLE OUTPUT
-----------------------------------------------
```

```
Preparing NY Style Cheese Pizza
Baking NY Style Cheese Pizza
Cutting NY Style Cheese Pizza
Boxing NY Style Cheese Pizza
Ordered: NY Style Cheese Pizza
Preparing Chicago Style Cheese Pizza
Baking Chicago Style Cheese Pizza
Cutting Chicago Style Cheese Pizza
Boxing Chicago Style Cheese Pizza
Ordered: Chicago Style Cheese Pizza
---------------------------------------------
*/
```

Slip 4b

Q2 Write a program to sense a finger when it is placed on the board Arduino.

```
#include <CapacitiveSensor.h>

const int touchPin = 2;              // Capacitive sensing pin
const int touchThreshold = 50;       // Sensitivity threshold

// Create capacitive sensor object (send pin = 0, receive pin = touchPin)
CapacitiveSensor touchSensor = CapacitiveSensor(0, touchPin);

void setup() {
  Serial.begin(9600);               // Start serial monitor
}

void loop() {
  long touchValue = touchSensor.capacitiveSensor(30);  // Read capacitive value
```

```
  if (touchValue > touchThreshold) {    // Check if finger is detected
    Serial.println("Finger detected!");
    delay(1000);                        // Delay to avoid repeated detection
  }
}
```

Slip 5
Slip 5a
Q1 Write a Java Program to implement Adapter pattern for Enumeration iterator
/*
Slip 5a
Q1 Write a Java Program to implement Adapter pattern for Enumeration to Iterator
*/

```
import java.util.*;

// Adapter class
class EnumerationIteratorAdapter<E> implements Iterator<E> {
   private Enumeration<E> enumeration;

   public EnumerationIteratorAdapter(Enumeration<E> enumeration) {
      this.enumeration = enumeration;
   }

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

   public E next() {
      return enumeration.nextElement();
   }

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

public class slip5a {
   public static void main(String[] args) {
      Vector<String> v = new Vector<>();
      v.add("Apple");
```

```
        v.add("Banana");
        v.add("Cherry");

        Enumeration<String> en = v.elements();
        Iterator<String> it = new EnumerationIteratorAdapter<>(en);

        System.out.println("Iterating using Adapter:");
        while (it.hasNext()) {
            System.out.println(it.next());
        }
    }
}

/*
------------------------------------------------
 TERMINAL COMMANDS (VS CODE)
------------------------------------------------
javac slip5a.java
java slip5a

------------------------------------------------
 SAMPLE OUTPUT
------------------------------------------------
Iterating using Adapter:
Apple
Banana
Cherry
------------------------------------------------
*/
```

Slip 5b

Q2 Write a program to connect with the available Wi-Fi using Arduino.

```
#include <SPI.h>
#include <WiFiNINA.h>

char ssid[] = "Wifi name";         // Your WiFi name
char pass[] = "secret password";   // Your WiFi password

int status = WL_IDLE_STATUS;       // WiFi status variable

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

  // Try connecting to WiFi
  while (status != WL_CONNECTED) {
    Serial.print("Connecting to ");
    Serial.println(ssid);

    status = WiFi.begin(ssid, pass);   // Start WiFi connection
    delay(5000);                       // Wait 5 seconds
  }

  // When connected
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());      // Print local IP
}

void loop() {
  // Nothing required here
}
```

Slip 6
Slip 6a
Q1 Write a Java Program to implement command pattern to test Remote Control
/*
Slip 6a
Q1 Write a Java Program to implement command pattern to test Remote Control
*/

```
import java.util.Scanner;

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

// Receiver class
class Light {
    private String name;
    public Light(String name) { this.name = name; }

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

// Concrete Commands
class LightOnCommand implements Command {
    private Light light;
    public LightOnCommand(Light light) { this.light = light; }
    public void execute() { light.on(); }
}

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

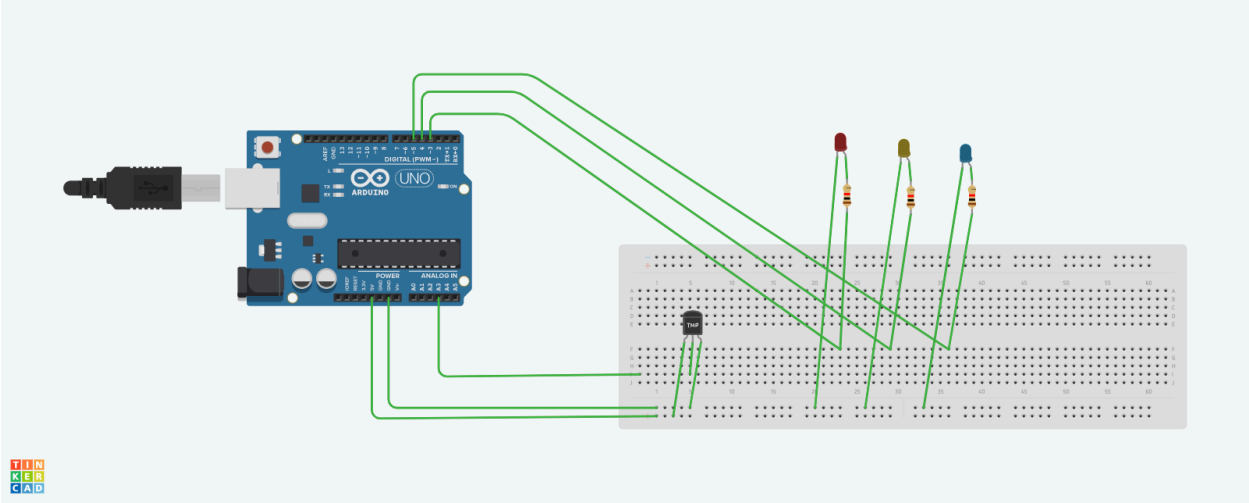
```
}

// Invoker class
class RemoteControl {
    private Command command;
    public void setCommand(Command command) { this.command = command; }
    public void pressButton() { command.execute(); }
}

public class slip6a {
    public static void main(String[] args) {
        Light livingRoom = new Light("Living Room");
        RemoteControl remote = new RemoteControl();
        Scanner sc = new Scanner(System.in);

        System.out.println("Press 1 to turn ON Living Room light, 0 to turn OFF, x to
exit");

        while (true) {
            String input = sc.nextLine();
            if (input.equals("x")) break;

            if (input.equals("1")) {
                remote.setCommand(new LightOnCommand(livingRoom));
                remote.pressButton();
            } else if (input.equals("0")) {
                remote.setCommand(new LightOffCommand(livingRoom));
                remote.pressButton();
            } else {
                System.out.println("Invalid input! Press 1, 0, or x.");
            }
        }

        sc.close();
        System.out.println("Program exited.");
```

```
    }
}

/*
-----------------------------------------------
 TERMINAL COMMANDS (VS CODE)
-----------------------------------------------
javac slip6a.java
java slip6a

-----------------------------------------------
 SAMPLE RUN
-----------------------------------------------
Press 1 to turn ON Living Room light, 0 to turn OFF, x to exit
1
Living Room Light is ON
0
Living Room Light is OFF
x
Program exited.
-----------------------------------------------
*/
```

Slip 6b
Q2 Write a program to get temperature notification using Arduino.

```
const int hot = 90;     // Hot temperature threshold (F)
const int cold = 60;    // Cold temperature threshold (F)

// Pin definitions
// A2 = Temperature sensor input
// 2 = Red LED
// 3 = Blue LED
// 4 = Green LED

void setup() {
  pinMode(A2, INPUT);    // Sensor pin
  pinMode(2, OUTPUT);    // Red LED
  pinMode(3, OUTPUT);    // Blue LED
  pinMode(4, OUTPUT);    // Green LED

  Serial.begin(9600);    // Start serial monitor
}

void loop() {
  int sensor = analogRead(A2);                  // Read sensor value
  float voltage = (sensor / 1024.0) * 5.0;      // Convert to voltage
  float tempC = (voltage - 0.5) * 100;          // Convert to Celsius
  float tempF = (tempC * 1.8) + 32;             // Convert °C → °F

  Serial.print("temp: ");
  Serial.print(tempF);

  // Cold Condition
  if (tempF < cold) {
    digitalWrite(2, HIGH);   // Red ON
```

```
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  Serial.println(" It's Cold.");
 }
 // Hot Condition
 else if (tempF >= hot) {
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);   // Green ON
  Serial.println(" It's Hot.");
 }
 // Fine Condition
 else {
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);   // Blue ON
  digitalWrite(4, LOW);
  Serial.println(" It's Fine.");
 }

 delay(400);
}
```



Arduino Uno/Genuino
Breadboard
3 resistors
3 RGB LEDs (any color)
Temperature Sensor [TMP36]

Slip 7
Slip 7a
Q1 Write a Java Program to implement undo command to test Ceiling fan.

```
/*
Slip 7a
Q1 Write a Java Program to implement undo command to test Ceiling fan
*/

import java.util.Scanner;

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

// Receiver class
class CeilingFan {
    private String location;
    private int speed;
    public static final int OFF = 0;
    public static final int LOW = 1;
    public static final int MEDIUM = 2;
    public static final int HIGH = 3;

    public CeilingFan(String location) {
        this.location = location;
        this.speed = OFF;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
        String state = switch(speed) {
            case HIGH -> "HIGH";
```

```
            case MEDIUM -> "MEDIUM";
            case LOW -> "LOW";
            default -> "OFF";
        };
        System.out.println(location + " Ceiling Fan is " + state);
    }

    public int getSpeed() { return speed; }
}

// Concrete Commands
class HighCommand implements Command {
    private CeilingFan fan;
    private int prevSpeed;
    public HighCommand(CeilingFan fan) { this.fan = fan; }
    public void execute() {
        prevSpeed = fan.getSpeed();
        fan.setSpeed(CeilingFan.HIGH);
    }
    public void undo() { fan.setSpeed(prevSpeed); }
}

class MediumCommand implements Command {
    private CeilingFan fan;
    private int prevSpeed;
    public MediumCommand(CeilingFan fan) { this.fan = fan; }
    public void execute() {
        prevSpeed = fan.getSpeed();
        fan.setSpeed(CeilingFan.MEDIUM);
    }
    public void undo() { fan.setSpeed(prevSpeed); }
}

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

```
    private int prevSpeed;
    public LowCommand(CeilingFan fan) { this.fan = fan; }
    public void execute() {
        prevSpeed = fan.getSpeed();
        fan.setSpeed(CeilingFan.LOW);
    }
    public void undo() { fan.setSpeed(prevSpeed); }
}

class OffCommand implements Command {
    private CeilingFan fan;
    private int prevSpeed;
    public OffCommand(CeilingFan fan) { this.fan = fan; }
    public void execute() {
        prevSpeed = fan.getSpeed();
        fan.setSpeed(CeilingFan.OFF);
    }
    public void undo() { fan.setSpeed(prevSpeed); }
}

// Invoker
class RemoteControl {
    private Command command;
    private Command lastCommand;

    public void setCommand(Command command) { this.command = command; }
    public void pressButton() {
        command.execute();
        lastCommand = command;
    }
    public void pressUndo() {
        if (lastCommand != null) lastCommand.undo();
    }
}
```

```
public class slip7a {
   public static void main(String[] args) {
      CeilingFan fan = new CeilingFan("Living Room");
      RemoteControl remote = new RemoteControl();
      Scanner sc = new Scanner(System.in);

      System.out.println("Press 3 for HIGH, 2 for MEDIUM, 1 for LOW, 0 for
OFF, u for UNDO, x to exit");

      while (true) {
         String input = sc.nextLine();
         if (input.equals("x")) break;

         switch(input) {
            case "3" -> { remote.setCommand(new HighCommand(fan));
remote.pressButton(); }
            case "2" -> { remote.setCommand(new MediumCommand(fan));
remote.pressButton(); }
            case "1" -> { remote.setCommand(new LowCommand(fan));
remote.pressButton(); }
            case "0" -> { remote.setCommand(new OffCommand(fan));
remote.pressButton(); }
            case "u" -> remote.pressUndo();
            default -> System.out.println("Invalid input! Press 0-3, u, or x.");
         }
      }

      sc.close();
      System.out.println("Program exited.");
   }
}

/*
-----------------------------------------------
 TERMINAL COMMANDS (VS CODE)
```

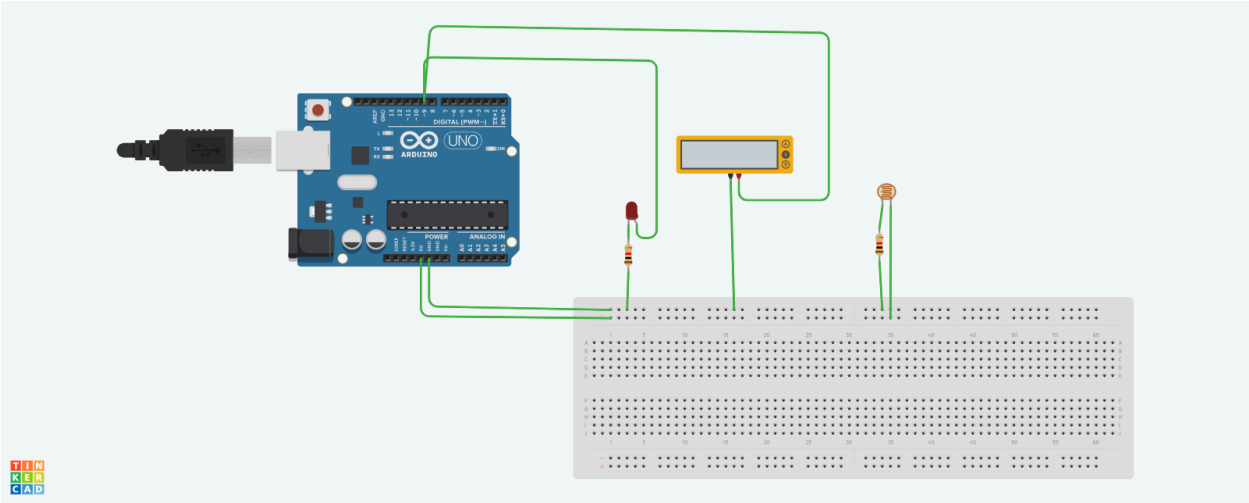
```
------------------------------------------------
javac slip7a.java
java slip7a

------------------------------------------------
 SAMPLE RUN
------------------------------------------------
Press 3 for HIGH, 2 for MEDIUM, 1 for LOW, 0 for OFF, u for UNDO, x to exit
3
Living Room Ceiling Fan is HIGH
2
Living Room Ceiling Fan is MEDIUM
u
Living Room Ceiling Fan is HIGH
0
Living Room Ceiling Fan is OFF
u
Living Room Ceiling Fan is HIGH
x
Program exited.
------------------------------------------------
*/
```

Slip 7b

Q2 Write a program for LDR to vary the light intensity of LED using Arduino.

```
int sensorValue = 0;   // store LDR value

void setup() {
  pinMode(A0, INPUT);     // LDR input pin
  Serial.begin(9600);     // begin serial monitor
  pinMode(9, OUTPUT);     // LED on PWM pin 9
```

```
}

void loop() {
  sensorValue = analogRead(A0);  // Read LDR value (0–1023)

  Serial.println(sensorValue);   // Print for testing

  // Map LDR value to LED brightness (0–255)
  analogWrite(9, map(sensorValue, 0, 1023, 0, 255));

  delay(100);   // Small delay for stability
}
```



Arduino UNO
Photoresistor
Multimeter
LDR(Photoregister)
Resistors Breadboard LED

Slip 8

Slip 8a
Q1 Write a Java Program to implement State Pattern for Gumball Machine.
Create instance variable that holds current state from there, we just need to handle all actions, behaviors and state transition that can happen

```
/*
Slip 8a
Q1 Write a Java Program to implement State Pattern for Gumball Machine
*/

import java.util.Scanner;

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

// Context class
class GumballMachine {
    State soldOutState;
    State noQuarterState;
    State hasQuarterState;
    State soldState;

    State state;
    int count;

    public GumballMachine(int numberGumballs) {
```

```
        soldOutState = new SoldOutState(this);
        noQuarterState = new NoQuarterState(this);
        hasQuarterState = new HasQuarterState(this);
        soldState = new SoldState(this);

        this.count = numberGumballs;
        if (numberGumballs > 0) state = noQuarterState;
        else state = soldOutState;
    }

    public void insertQuarter() { state.insertQuarter(); }
    public void ejectQuarter() { state.ejectQuarter(); }
    public void turnCrank() { state.turnCrank(); state.dispense(); }
    void setState(State state) { this.state = state; }
    void releaseBall() {
        if (count > 0) {
            System.out.println("A gumball comes rolling out...");
            count--;
        }
    }
    public int getCount() { return count; }

    public State getSoldOutState() { return soldOutState; }
    public State getNoQuarterState() { return noQuarterState; }
    public State getHasQuarterState() { return hasQuarterState; }
    public State getSoldState() { return soldState; }
}

// Concrete States
class SoldOutState implements State {
    GumballMachine machine;
    public SoldOutState(GumballMachine m) { machine = m; }
    public void insertQuarter() { System.out.println("Machine is sold out"); }
```

```
    public void ejectQuarter() { System.out.println("No quarter to eject"); }
    public void turnCrank() { System.out.println("Turned crank but no gumball");
}
    public void dispense() { System.out.println("No gumball dispensed"); }
}

class NoQuarterState implements State {
    GumballMachine machine;
    public NoQuarterState(GumballMachine m) { machine = m; }
    public void insertQuarter() {
        System.out.println("You inserted a quarter");
        machine.setState(machine.getHasQuarterState());
    }
    public void ejectQuarter() { System.out.println("No quarter to eject"); }
    public void turnCrank() { System.out.println("You turned but no quarter"); }
    public void dispense() { System.out.println("You need to pay first"); }
}

class HasQuarterState implements State {
    GumballMachine machine;
    public HasQuarterState(GumballMachine m) { machine = m; }
    public void insertQuarter() { System.out.println("Quarter already inserted"); }
    public void ejectQuarter() {
        System.out.println("Quarter returned");
        machine.setState(machine.getNoQuarterState());
    }
    public void turnCrank() {
        System.out.println("You turned...");
        machine.setState(machine.getSoldState());
    }
    public void dispense() { System.out.println("No gumball dispensed"); }
}
```

```
class SoldState implements State {
    GumballMachine machine;
    public SoldState(GumballMachine m) { machine = m; }
    public void insertQuarter() { System.out.println("Wait, we're already giving
you a gumball"); }
    public void ejectQuarter() { System.out.println("Can't eject, 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 machine.setState(machine.getSoldOutState());
    }
}

// Test class
public class slip8a {
    public static void main(String[] args) {
        GumballMachine machine = new GumballMachine(3);
        Scanner sc = new Scanner(System.in);
        System.out.println("Commands: i = insert quarter, e = eject quarter, t = turn
crank, x = exit");

        while (true) {
            String input = sc.nextLine();
            if (input.equals("x")) break;

            switch (input) {
                case "i" -> machine.insertQuarter();
                case "e" -> machine.ejectQuarter();
                case "t" -> machine.turnCrank();
                default -> System.out.println("Invalid command!");
```

```
            }
        }

        sc.close();
        System.out.println("Program exited.");
    }
}

/*
-----------------------------------------------
 TERMINAL COMMANDS (VS CODE)
-----------------------------------------------
javac slip8a.java
java slip8a

-----------------------------------------------
 SAMPLE RUN
-----------------------------------------------
Commands: i = insert quarter, e = eject quarter, t = turn crank, x = exit
i
You inserted a quarter
t
You turned...
A gumball comes rolling out...
t
You turned but no quarter
i
You inserted a quarter
e
Quarter returned
x
Program exited.
-----------------------------------------------
```

*/

Slip 8b
Q2 Start Raspberry Pi and execute various Linux commands in the command terminal window: **ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping** etc.

Here are the **main Linux commands** you must execute on the Raspberry Pi with their simple purpose:

- **ls** – List files and folders
- **cd** – Change directory
- **touch** – Create an empty file
- **mv** – Move or rename a file
- **rm** – Remove/delete a file
- **man** – Show manual/help for a command
- **mkdir** – Create a new folder
- **rmdir** – Remove an empty folder
- **tar** – Create/extract archive files
- **gzip** – Compress or decompress files

- **cat** – View contents of a file
- **more** – View long files page-by-page
- **less** – View files with scroll up/down
- **ps** – Show running processes
- **sudo** – Run command as administrator
- **cron** – Schedule automated tasks
- **chown** – Change file owner
- **chgrp** – Change file group
- **ping** – Test network connection

Slip 9
Slip 9a
Q1 Design simple HR Application using Spring Framework

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.*;
import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

// Configuration class
@Configuration
@ComponentScan
class AppConfig {}

// Employee class
class Employee {
    private String name;
    private String role;

    public Employee(String name, String role) {
        this.name = name;
        this.role = role;
    }

    public String getName() { return name; }
    public String getRole() { return role; }
}

// HR Service
@Component
class HRService {
    private List<Employee> employees = new ArrayList<>();
```

```
    public HRService() {
        // default employees
        employees.add(new Employee("Ram", "Developer"));
        employees.add(new Employee("Vaibhav", "Tester"));
    }

    public void addEmployee(String name, String role) {
        employees.add(new Employee(name, role));
    }

    public List<Employee> getAllEmployees() {
        return employees;
    }
}

// Main class
public class slip9a {
    public static void main(String[] args) {
        ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
        HRService hr = context.getBean(HRService.class);

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter employee name: ");
        String name = sc.nextLine();
        System.out.print("Enter employee role: ");
        String role = sc.nextLine();

        hr.addEmployee(name, role);

        System.out.println("\n=== HR Employee List ===");
        for(Employee e : hr.getAllEmployees()) {
            System.out.println("Name: " + e.getName() + ", Role: " + e.getRole());
        }
```

```
        sc.close();
    }
}


/*
Example Input (from terminal):
Enter employee name: Sita
Enter employee role: Manager

Example Output:
=== HR Employee List ===
Name: Ram, Role: Developer
*/

/*
IMPORTANT:
You need Spring framework JARs in a folder named "libs" in the same directory.
Required JARs: spring-core, spring-context, spring-beans, spring-expression

Windows:
javac -cp "libs/*" slip9a.java
java -cp ".;libs/*" slip9a

Linux/Mac:
javac -cp "libs/*" slip9a.java
java -cp ".:libs/*" slip9a
*/
```

Slip 9b

Q2 Write python programs on Raspberry Pi to perform the following:

Read your name and print a “Hello” message with the name.
Read two numbers and print their sum, difference, product, and division.
Count the number of words and characters of a given string.
Find the area of a given shape (rectangle, triangle, and circle) by reading the shape and appropriate values from standard input.

```
# ------------------------------
# 1. Read your name and print Hello message
# ------------------------------
name = input("Enter your name: ")
print("Hello", name + "! Welcome to Raspberry Pi.")

# ------------------------------
# 2. Read two numbers and print sum, difference, product, division
# ------------------------------
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

print("Sum =", num1 + num2)
print("Difference =", num1 - num2)
print("Product =", num1 * num2)

if num2 != 0:
    print("Division =", num1 / num2)
else:
    print("Division = Not possible (Cannot divide by zero)")

# ------------------------------
# 3. Word and character count of a given string
# ------------------------------
text = input("Enter a sentence: ")

word_count = len(text.split())
char_count = len(text)
```

```
print("Word Count:", word_count)
print("Character Count:", char_count)

# ------------------------------
# 4. Area of a given shape
# rectangle, triangle, circle
# ------------------------------
shape = input("Enter shape (rectangle/triangle/circle): ").lower()

if shape == "rectangle":
    length = float(input("Enter length: "))
    width = float(input("Enter width: "))
    area = length * width
    print("Area of Rectangle =", area)

elif shape == "triangle":
    base = float(input("Enter base: "))
    height = float(input("Enter height: "))
    area = 0.5 * base * height
    print("Area of Triangle =", area)

elif shape == "circle":
    radius = float(input("Enter radius: "))
    area = 3.14159 * radius * radius
    print("Area of Circle =", area)

else:
    print("Invalid shape entered!")
```

Slip 10

Slip 10a

Q1 Write a Java Program to implement Strategy Pattern for Duck Behavior. Create instance variable that holds current state of Duck from there, we just need to handle all Flying Behaviors and Quack Behavior

/*

Slip 10a
Q1 Write a Java Program to implement Strategy Pattern for Duck Behavior.
*/

```
// Strategy Interfaces
interface FlyBehavior {
    void fly();
}

interface QuackBehavior {
    void quack();
}

// Concrete Fly Behaviors
class FlyWithWings implements FlyBehavior {
    public void fly() {
        System.out.println("Flying with wings!");
    }
}

class FlyNoWay implements FlyBehavior {
    public void fly() {
        System.out.println("Can't fly!");
    }
}

// Concrete Quack Behaviors
class Quack implements QuackBehavior {
    public void quack() {
        System.out.println("Quack Quack!");
    }
}
```

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

// Duck class
class Duck {
    private FlyBehavior flyBehavior;
    private QuackBehavior quackBehavior;

    public Duck(FlyBehavior f, QuackBehavior q) {
        flyBehavior = f;
        quackBehavior = q;
    }

    public void performFly() {
        flyBehavior.fly();
    }

    public void performQuack() {
        quackBehavior.quack();
    }

    public void setFlyBehavior(FlyBehavior f) { flyBehavior = f; }
    public void setQuackBehavior(QuackBehavior q) { quackBehavior = q; }
}

// Main class
public class slip10a {
    public static void main(String[] args) {
        Duck mallard = new Duck(new FlyWithWings(), new Quack());
        Duck rubber = new Duck(new FlyNoWay(), new MuteQuack());
```

```
        System.out.println("Mallard Duck:");
        mallard.performFly();
        mallard.performQuack();

        System.out.println("\nRubber Duck:");
        rubber.performFly();
        rubber.performQuack();

        // Change behavior at runtime
        System.out.println("\nRubber Duck learns to fly!");
        rubber.setFlyBehavior(new FlyWithWings());
        rubber.performFly();
    }
}

/*
------------------------------------------------
 VS CODE TERMINAL COMMANDS
------------------------------------------------
javac slip10a.java
java slip10a

------------------------------------------------
 SAMPLE OUTPUT
------------------------------------------------
Mallard Duck:
Flying with wings!
Quack Quack!

Rubber Duck:
Can't fly!
<< Silence >>
```

Rubber Duck learns to fly!
Flying with wings!
---
*/

Slip 10b
Q2 Write python programs on Pi for the following:
Print a name 'n' times, where name and n are read from standard input, using for and while loops.
Handle "Divide by Zero" exception.
Print current time 10 times, with an interval of 10 seconds.
Read a file line by line and print the word count of each line.

```
import time
from datetime import datetime

# ------------------------------------------------
# 1. Print a name 'n' times using for loop and while loop
# ------------------------------------------------
name = input("Enter your name: ")
n = int(input("Enter how many times to print: "))

print("\nUsing FOR loop:")
for i in range(n):
    print(name)

print("\nUsing WHILE loop:")
count = 0
while count < n:
```

```
    print(name)
    count += 1


# ------------------------------------------------
# 2. Handle Divided by Zero Exception
# ------------------------------------------------
print("\nDivision Program with Exception Handling:")

a = float(input("Enter numerator: "))
b = float(input("Enter denominator: "))

try:
    result = a / b
    print("Result =", result)
except ZeroDivisionError:
    print("Error: Cannot divide by zero!")


# ------------------------------------------------
# 3. Print current time for 10 times with 10-second interval
# ------------------------------------------------
print("\nPrinting current time 10 times with 10-second interval:")

for _ in range(10):
    print("Current Time:", datetime.now().strftime("%H:%M:%S"))
    time.sleep(10)   # 10-second delay


# ------------------------------------------------
# 4. Read a file line by line and print word count of each line
# ------------------------------------------------
print("\nWord Count of File Lines:")
```

```
file_name = input("Enter file name to read: ")

try:
    with open(file_name, "r") as file:
        line_number = 1
        for line in file:
            words = line.split()
            print(f"Line {line_number}: {len(words)} words")
            line_number += 1
except FileNotFoundError:
    print("Error: File not found!")
```

Slip 11

Slip 11a
Q1 Write a java program to implement Adapter pattern to design Heart Model to Beat Model

```
/*
Slip 11a
Q1 Write a Java Program to implement Adapter pattern to design Heart Model to Beat Model
*/

// Adaptee: HeartModel
class HeartModel {
    private int heartRate;

    public HeartModel() {
        heartRate = 72; // default beats per minute
    }

    public void pump() {
        System.out.println("Heart is pumping at " + heartRate + " beats per minute");
    }

    public void setHeartRate(int rate) {
        heartRate = rate;
    }
```

```
    public int getHeartRate() {
        return heartRate;
    }
}

// Target interface: BeatModel
interface BeatModel {
    void startBeating();
    void setRate(int rate);
    int getRate();
}

// Adapter class
class HeartAdapter implements BeatModel {
    private HeartModel heart;

    public HeartAdapter(HeartModel heart) {
        this.heart = heart;
    }

    @Override
    public void startBeating() {
        heart.pump();
    }

    @Override
    public void setRate(int rate) {
        heart.setHeartRate(rate);
    }

    @Override
    public int getRate() {
        return heart.getHeartRate();
```

```
    }
}

// Main class
public class slip11a {
    public static void main(String[] args) {
        HeartModel heart = new HeartModel();
        BeatModel beatHeart = new HeartAdapter(heart);

        System.out.println("Initial Heart Beat:");
        beatHeart.startBeating();

        System.out.println("\nChanging Heart Rate to 90 bpm:");
        beatHeart.setRate(90);
        beatHeart.startBeating();

        System.out.println("\nCurrent Heart Rate: " + beatHeart.getRate() + "
bpm");
    }
}

/*
------------------------------------------------
 VS CODE TERMINAL COMMANDS
------------------------------------------------
javac slip11a.java
java slip11a

------------------------------------------------
 SAMPLE OUTPUT
------------------------------------------------
Initial Heart Beat:
Heart is pumping at 72 beats per minute
```

Changing Heart Rate to 90 bpm:
Heart is pumping at 90 beats per minute

Current Heart Rate: 90 bpm
---
*/

Slip 11b
Q2 Run some python programs on Pi like:
Light an LED through Python program
Get input from two switches and switch on corresponding LEDs
Flash an LED at a given ON time and OFF time cycle, where the two times are taken from a file

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Pin setup (common for all)
LED_MAIN = 18

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

```
# ------------------------------
# Program 1: Light a single LED
# ------------------------------
def light_single_led():
    GPIO.setup(LED_MAIN, GPIO.OUT)

    print("\n--- PROGRAM 1: Light LED ---")
    print("LED ON for 5 seconds...")
    GPIO.output(LED_MAIN, GPIO.HIGH)
    time.sleep(5)

    print("LED OFF")
    GPIO.output(LED_MAIN, GPIO.LOW)



# ------------------------------------------------------
# Program 2: Two switches control two corresponding LEDs
# ------------------------------------------------------
def switch_control_leds():
    print("\n--- PROGRAM 2: Switches control LEDs ---")

    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)

    print("Press CTRL+C to exit")

    try:
        while True:
            # Switch 1
            if GPIO.input(SW1) == GPIO.LOW:
                GPIO.output(LED1, GPIO.HIGH)
```

```
            else:
                GPIO.output(LED1, GPIO.LOW)

            # Switch 2
            if GPIO.input(SW2) == GPIO.LOW:
                GPIO.output(LED2, GPIO.HIGH)
            else:
                GPIO.output(LED2, GPIO.LOW)

            time.sleep(0.1)

    except KeyboardInterrupt:
        GPIO.output(LED1, GPIO.LOW)
        GPIO.output(LED2, GPIO.LOW)
        print("\nProgram exited")


# ----------------------------------------------------
# Program 3: Flash LED with timing from file
# ----------------------------------------------------
def flash_led_from_file():
    print("\n--- PROGRAM 3: Flash LED using ON/OFF times from file ---")

    GPIO.setup(LED_MAIN, GPIO.OUT)

    # Read from file
    try:
        with open("timing.txt", "r") as file:
            on_time = float(file.readline().strip())
            off_time = float(file.readline().strip())
    except:
        print("Error: timing.txt not found or invalid format")
        return
```

```
    print(f"Flashing LED → ON={on_time}s, OFF={off_time}s")
    print("Press CTRL+C to stop")

    try:
        while True:
            GPIO.output(LED_MAIN, GPIO.HIGH)
            time.sleep(on_time)

            GPIO.output(LED_MAIN, GPIO.LOW)
            time.sleep(off_time)

    except KeyboardInterrupt:
        GPIO.output(LED_MAIN, GPIO.LOW)
        print("\nStopped flashing LED")


# ------------------------------
# MAIN MENU
# ------------------------------
def main_menu():
    while True:
        print("\n========== SLIP 11 - Q2 ==========")
        print("1. Light an LED")
        print("2. Control LEDs using two switches")
        print("3. Flash LED using ON/OFF time from file")
        print("4. Exit")
        print("====================================")

        ch = input("Enter your choice: ")

        if ch == "1":
            light_single_led()
```

```
        elif ch == "2":
            switch_control_leds()
        elif ch == "3":
            flash_led_from_file()
        elif ch == "4":
            print("Exiting...")
            GPIO.cleanup()
            break
        else:
            print("Invalid choice! Try again.")


# Run the menu
try:
    main_menu()
except:
    GPIO.cleanup()
```

Slip 12

Slip 12a

Q1 Write a Java Program to implement Decorator Pattern for interface Car to define the assemble() method and then decorate it to Sports car and Luxury Car

[15 M]
/*

Slip 12a

Q1 Write a Java Program to implement Decorator Pattern for interface Car to define the assemble() method and then decorate it to Sports Car and Luxury Car

*/

```
// Component Interface
interface Car {
    void assemble();
}

// Concrete Component
class BasicCar implements Car {
    @Override
    public void assemble() {
        System.out.print("Basic Car");
    }
}

// Decorator Class
class CarDecorator implements Car {
```

```
    protected Car car;

    public CarDecorator(Car car) {
        this.car = car;
    }

    @Override
    public void assemble() {
        car.assemble();
    }
}

// Concrete Decorators
class SportsCar extends CarDecorator {
    public SportsCar(Car car) { super(car); }

    @Override
    public void assemble() {
        super.assemble();
        System.out.print(" + Sports Features");
    }
```

```
}

class LuxuryCar extends CarDecorator {
    public LuxuryCar(Car car) { super(car); }

    @Override
    public void assemble() {
        super.assemble();
        System.out.print(" + Luxury Features");
    }
}

// Main class
public class slip12a {
    public static void main(String[] args) {
        Car sportsCar = new SportsCar(new BasicCar());
        Car luxuryCar = new LuxuryCar(new BasicCar());

        System.out.println("Sports Car Configuration:");
        sportsCar.assemble();
        System.out.println();
```

```
        System.out.println("Luxury Car Configuration:");
        luxuryCar.assemble();
        System.out.println();
    }
}

/*
-----------------------------------------------
 VS CODE TERMINAL COMMANDS
-----------------------------------------------
javac slip12a.java
java slip12a

-----------------------------------------------
 SAMPLE OUTPUT
-----------------------------------------------
Sports Car Configuration:
Basic Car + Sports Features
Luxury Car Configuration:
Basic Car + Luxury Features
```

```
-----------------------------------------------
*/
```

Slip 12b

Q2 Write a program to sense the available networks using Arduino

```
#include <WiFiNINA.h>
void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Scanning available WiFi networks...");
}
void loop() {
  int networks = WiFi.scanNetworks();
  Serial.println("-----------------------------------------------");
  Serial.print("Number of networks found: ");
  Serial.println(networks);
  Serial.println("-----------------------------------------------");
  for (int i = 0; i < networks; i++) {
    Serial.print(i + 1);
```

```
    Serial.print(". ");
    Serial.print(WiFi.SSID(i));              // Network name
    Serial.print(" | Signal Strength (RSSI): ");
    Serial.print(WiFi.RSSI(i));              // Signal power
    Serial.print(" dBm | Encryption: ");
    Serial.println(WiFi.encryptionType(i));  // Encryption type
  }
  Serial.println("-----------------------------------------------\n");
  delay(5000);  // Scan every 5 seconds
}
```

Slip 13

Slip 13a

Q1 Write a Java Program to implement an Adapter design pattern in mobile charger. Define two classes — Volt (to measure volts) and Socket (producing constant volts of 120V). Build an adapter that can produce 3 volts, 12 volts and default 120 volts. Implements Adapter pattern using Class Adapter [15 M]

```
/*
Slip 13a
Q1 Write a Java Program to implement an Adapter design pattern in mobile charger.
  Define two classes — Volt (to measure volts) and Socket (producing constant volts of 120V).
  Build an adapter that can produce 3 volts, 12 volts and default 120 volts.
  Implements Adapter pattern using Class Adapter.
*/

// Volt class
class Volt {
    private int volts;

    public Volt(int v) { volts = v; }

    public int getVolts() { return volts; }
}

// Socket class (Adaptee)
class Socket {
    public Volt getVolt() {
```

```
        return new Volt(120); // default socket voltage
    }
}

// Adapter class using inheritance (Class Adapter)
class SocketAdapter extends Socket {
    public Volt get3Volt() { return convertVolt(getVolt(), 40); }   // 120/40 = 3V
    public Volt get12Volt() { return convertVolt(getVolt(), 10); }  // 120/10 = 12V
    public Volt get120Volt() { return getVolt(); }                  // default

    private Volt convertVolt(Volt v, int divisor) {
        return new Volt(v.getVolts() / divisor);
    }
}

// Main class
public class slip13a {
    public static void main(String[] args) {
        SocketAdapter adapter = new SocketAdapter();
```

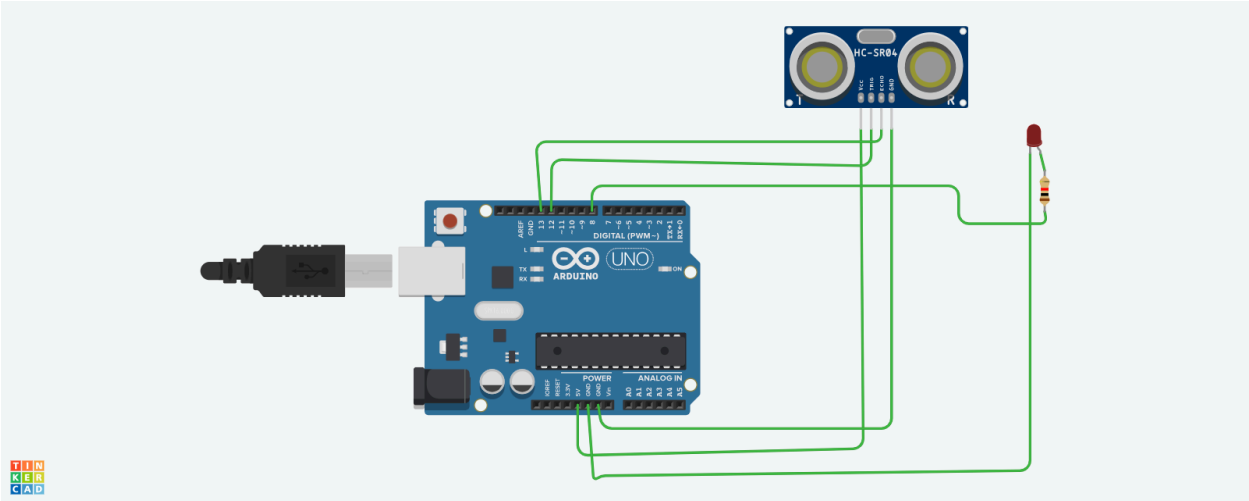
```
        System.out.println("Default Voltage: " + adapter.get120Volt().getVolts() +
"V");
        System.out.println("3 Volt: " + adapter.get3Volt().getVolts() + "V");
        System.out.println("12 Volt: " + adapter.get12Volt().getVolts() + "V");
    }
}

/*
-----------------------------------------------
 VS CODE TERMINAL COMMANDS
-----------------------------------------------
javac slip13a.java
java slip13a

-----------------------------------------------
 SAMPLE OUTPUT
-----------------------------------------------
Default Voltage: 120V
3 Volt: 3V
12 Volt: 12V
-----------------------------------------------
```

*/

Slip 13b
Q2 Write a program to measure the distance using ultrasonic sensor and make LED blink using Arduino.

```
// Pin definitions
const int trig = 12;
const int echo = 13;
const int LED1 = 8;

int duration = 0;
int distance = 0;

void setup() {
  pinMode(trig, OUTPUT);   // Trigger pin sends ultrasonic pulse
  pinMode(echo, INPUT);    // Echo pin receives reflected pulse
  pinMode(LED1, OUTPUT);   // LED output

  Serial.begin(9600);      // Start serial monitor
}

void loop() {

  // Send a HIGH pulse to trigger ultrasonic
  digitalWrite(trig, HIGH);
  delayMicroseconds(1000);   // Important: pulse duration for sensor
  digitalWrite(trig, LOW);

  // Read how long echo pin stayed HIGH (time of reflection)
```

```
  duration = pulseIn(echo, HIGH);
  // Convert time into distance in cm
  distance = (duration / 2) / 28.5;   // Speed of sound based formula
  Serial.println(distance);   // Print distance on Serial Monitor
  // Check if object is close (≤ 7 cm)
  if (distance <= 7) {
    digitalWrite(LED1, HIGH);   // LED ON when object is close
  }
  else {
    digitalWrite(LED1, LOW);    // LED OFF when object is far
  }
}
```



Arduino UNO,
Ultrasonic,
Register,
LED

Slip 14

Slip 14a

Q1 Write a Java Program to implement Command Design Pattern for Command Interface with execute() . Use this to create variety of commands for LightOnCommand, LightOffCommand, GarageDoorUpCommand, StereoOnWithCDComman. [15 M]

```
/*
Slip 14a
Q1 Write a Java Program to implement Command Design Pattern for Command Interface with execute().
   Use this to create variety of commands for LightOnCommand, LightOffCommand, GarageDoorUpCommand, MusicPlayerOnCommand.
   Added input-based actions for interactive use.
*/

import java.util.Scanner;

// Command Interface
interface Command {
    void execute();
}

// Receiver Classes
class Light {
```

```
    private String location;
    public Light(String location) { this.location = location; }
    public void on() { System.out.println(location + " Light is ON"); }
    public void off() { System.out.println(location + " Light is OFF"); }
}

class GarageDoor {
    public void up() { System.out.println("Garage Door is OPEN"); }
    public void down() { System.out.println("Garage Door is CLOSED"); }
}

class MusicPlayer {
    public void on() { System.out.println("Music Player is ON"); }
    public void off() { System.out.println("Music Player is OFF"); }
    public void playCD() { System.out.println("Music Player is playing CD"); }
}

// Concrete Commands
class LightOnCommand implements Command {
    private Light light;
    public LightOnCommand(Light light) { this.light = light; }
```

```
    public void execute() { light.on(); }
}

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

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

class MusicPlayerOnCommand implements Command {
    private MusicPlayer player;
    public MusicPlayerOnCommand(MusicPlayer player) { this.player = player;
}
    public void execute() {
        player.on();
```

```
        player.playCD();
    }
}

// Invoker
class RemoteControl {
    private Command slot;
    public void setCommand(Command command) { slot = command; }
    public void pressButton() {
        if(slot != null) slot.execute();
        else System.out.println("No command assigned!");
    }
}

// Main Class
public class slip14a {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        RemoteControl remote = new RemoteControl();

        Light livingRoomLight = new Light("Living Room");
```

```
        GarageDoor garageDoor = new GarageDoor();
        MusicPlayer musicPlayer = new MusicPlayer();

        while(true) {
            System.out.println("\n--- Remote Control ---");
            System.out.println("1. Turn ON Living Room Light");
            System.out.println("2. Turn OFF Living Room Light");
            System.out.println("3. Open Garage Door");
            System.out.println("4. Turn ON Music Player and Play CD");
            System.out.println("0. Exit");
            System.out.print("Enter choice: ");

            int choice = sc.nextInt();

            switch(choice) {
                case 1: remote.setCommand(new
LightOnCommand(livingRoomLight)); break;
                case 2: remote.setCommand(new
LightOffCommand(livingRoomLight)); break;
                case 3: remote.setCommand(new
GarageDoorUpCommand(garageDoor)); break;
```

```
            case 4: remote.setCommand(new
MusicPlayerOnCommand(musicPlayer)); break;
            case 0: System.out.println("Exiting..."); sc.close(); return;
            default: System.out.println("Invalid choice!"); continue;
        }
        remote.pressButton();
    }
  }
}

/*
-----------------------------------------------
 VS CODE TERMINAL COMMANDS
-----------------------------------------------
javac slip14a.java
java slip14a

-----------------------------------------------
 SAMPLE INTERACTIVE OUTPUT
-----------------------------------------------
--- Remote Control ---
```

1. Turn ON Living Room Light

2. Turn OFF Living Room Light

3. Open Garage Door

4. Turn ON Music Player and Play CD

0. Exit

Enter choice: 1

Living Room Light is ON

Enter choice: 4

Music Player is ON

Music Player is playing CD

---

*/

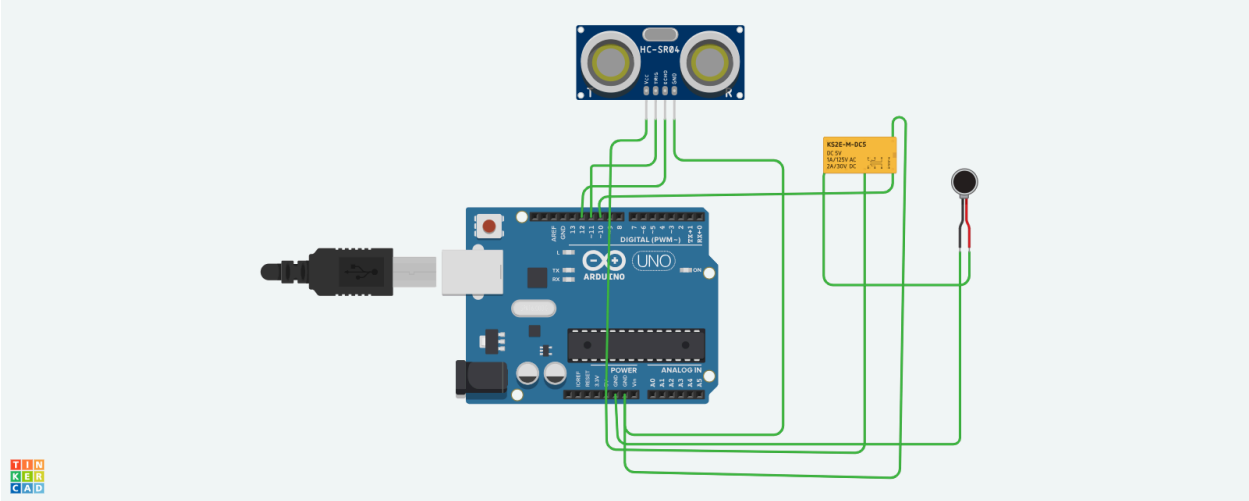
Slip 14b

Q2 Write a program to detects the vibration of an object with sensor using Arduino.

// Pin definitions

```
const int UltrasonikTrig1 = 11;
const int UltrasonikEcho1 = 12;
const int Indikator = 10;

// Variables
long duration1;
long cm1;

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

  pinMode(UltrasonikTrig1, OUTPUT);   // Trigger pin for ultrasonic
  pinMode(UltrasonikEcho1, INPUT);    // Echo pin for ultrasonic
  pinMode(Indikator, OUTPUT);         // LED or indicator output
}

void loop() {
  // Send pulse to ultrasonic sensor
  digitalWrite(UltrasonikTrig1, LOW);
  delayMicroseconds(2);

  digitalWrite(UltrasonikTrig1, HIGH);
  delayMicroseconds(10);
  digitalWrite(UltrasonikTrig1, LOW);

  // Read the echo return time
  duration1 = pulseIn(UltrasonikEcho1, HIGH);

  // Convert time to centimeters
  cm1 = microsecondsKeCenti(duration1);

  // Print distance value
  Serial.print("Sensor 1: ");
```

```
  Serial.print(cm1);
  Serial.print(" cm  ||  ");

  // If distance is less than or equal to 100 cm
  if (cm1 <= 100) {
    digitalWrite(Indikator, HIGH);   // Object close → Indicator ON
  } else {
    digitalWrite(Indikator, LOW);    // Object far → Indicator OFF
    Serial.print(" Safe");
  }

  Serial.println(" ||");
  delay(100);
}

// Convert microseconds to centimeters
long microsecondsKeCenti(long microseconds) {
  return microseconds / 29 / 2;
}
```



ultrasonic distance sensor
vibration motor

arduino uno r3
Relay DPDT

Slip 15

Slip 15a

Q1 Write a Java Program to implement Facade Design Pattern for Home Theater
/*

Slip 15a

Q1 Write a Java Program to implement Facade Design Pattern for Home Theater.

Added input-based interactive control.

*/

```
import java.util.Scanner;


// Subsystems
class Amplifier {
    public void on() { System.out.println("Amplifier is ON"); }
    public void off() { System.out.println("Amplifier is OFF"); }
    public void setVolume(int level) { System.out.println("Amplifier volume set
to " + level); }
}


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


class DVDPlayer {
    public void on() { System.out.println("DVD Player is ON"); }
    public void off() { System.out.println("DVD Player is OFF"); }
    public void play(String movie) { System.out.println("Playing movie: " +
movie); }
}
```

```
class Projector {
    public void on() { System.out.println("Projector is ON"); }
    public void off() { System.out.println("Projector is OFF"); }
    public void wideScreenMode() { System.out.println("Projector in widescreen mode"); }
}

// Facade
class HomeTheaterFacade {
    private Amplifier amp;
    private Tuner tuner;
    private DVDPlayer dvd;
    private Projector projector;

    public HomeTheaterFacade(Amplifier amp, Tuner tuner, DVDPlayer dvd, Projector projector) {
        this.amp = amp;
        this.tuner = tuner;
        this.dvd = dvd;
        this.projector = projector;
    }
```

```
    public void watchMovie(String movie) {
        System.out.println("\nGet ready to watch a movie...");
        amp.on();
        amp.setVolume(5);
        tuner.on();
        projector.on();
        projector.wideScreenMode();
        dvd.on();
        dvd.play(movie);
    }

    public void endMovie() {
        System.out.println("\nShutting down movie theater...");
        dvd.off();
        projector.off();
        tuner.off();
        amp.off();
    }
}
```

```
// Main Class
public class slip15a {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        Amplifier amp = new Amplifier();
        Tuner tuner = new Tuner();
        DVDPlayer dvd = new DVDPlayer();
        Projector projector = new Projector();

        HomeTheaterFacade homeTheater = new HomeTheaterFacade(amp, tuner,
dvd, projector);

        while(true) {
            System.out.println("\n--- Home Theater Control ---");
            System.out.println("1. Watch Movie");
            System.out.println("2. End Movie");
            System.out.println("0. Exit");
            System.out.print("Enter your choice: ");

            int choice = sc.nextInt();
```

```
            sc.nextLine(); // Consume newline

            switch(choice) {
                case 1:
                    System.out.print("Enter movie name: ");
                    String movie = sc.nextLine();
                    homeTheater.watchMovie(movie);
                    break;
                case 2:
                    homeTheater.endMovie();
                    break;
                case 0:
                    System.out.println("Exiting Home Theater...");
                    sc.close();
                    return;
                default:
                    System.out.println("Invalid choice!");
            }
        }
    }
}
```

/*

VS CODE TERMINAL COMMANDS

---

javac slip15a.java

java slip15a

SAMPLE INTERACTIVE OUTPUT

--- Home Theater Control ---

1. Watch Movie

2. End Movie

0. Exit

Enter your choice: 1

Enter movie name: Avengers

Get ready to watch a movie...

Amplifier is ON

Amplifier volume set to 5

Tuner is ON

Projector is ON

Projector in widescreen mode

DVD Player is ON

Playing movie: Avengers

--- Home Theater Control ---

1. Watch Movie

2. End Movie

0. Exit

Enter your choice: 2

Shutting down movie theater...

DVD Player is OFF

Projector is OFF

Tuner is OFF

Amplifier is OFF

---

*/

Slip 15b

Q2 Write a program to sense a finger when it is placed on the board Arduino.

```
#include <CapacitiveSensor.h>

const int touchPin = 2;              // Capacitive sensing pin
const int touchThreshold = 50;       // Sensitivity threshold

// Create capacitive sensor object (send pin = 0, receive pin = touchPin)
CapacitiveSensor touchSensor = CapacitiveSensor(0, touchPin);

void setup() {
  Serial.begin(9600);                // Start serial monitor
}

void loop() {
  long touchValue = touchSensor.capacitiveSensor(30);  // Read capacitive value

  if (touchValue > touchThreshold) {    // Check if finger is detected
    Serial.println("Finger detected!");
    delay(1000);                        // Delay to avoid repeated detection
  }
}
```

Slip 16

Slip 16a

Q1 Write a Java Program to implement Observer Design Pattern for number conversion. Accept a number in Decimal form and represent it in Hexadecimal, Octal and Binary. Change the Number and it reflects in other forms also

/*

Slip 16a

Q1 Write a Java Program to implement Observer Design Pattern for number conversion.

Accept a number in Decimal form and represent it in Hexadecimal, Octal, and Binary.

Changing the number updates all representations automatically.

*/

```
import java.util.*;

// Subject Interface
interface Subject {
    void addObserver(Observer o);
    void removeObserver(Observer o);
    void notifyObservers();
}

// Observer Interface
interface Observer {
    void update(int number);
}

// Concrete Subject
class NumberSubject implements Subject {
    private List<Observer> observers = new ArrayList<>();
    private int number;

    public void setNumber(int number) {
        this.number = number;
```

```
        notifyObservers();
    }

    public int getNumber() {
        return number;
    }

    public void addObserver(Observer o) { observers.add(o); }
    public void removeObserver(Observer o) { observers.remove(o); }
    public void notifyObservers() {
        for(Observer o : observers) {
            o.update(number);
        }
    }
}

// Concrete Observers
class HexObserver implements Observer {
    public void update(int number) {
        System.out.println("Hexadecimal: " +
Integer.toHexString(number).toUpperCase());
```

```
    }
}

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

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

// Main Class
public class slip16a {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        NumberSubject subject = new NumberSubject();
```

```
        subject.addObserver(new HexObserver());
        subject.addObserver(new OctObserver());
        subject.addObserver(new BinObserver());

        while(true) {
            System.out.print("\nEnter a decimal number (or -1 to exit): ");
            int num = sc.nextInt();
            if(num == -1) {
                System.out.println("Exiting...");
                break;
            }
            subject.setNumber(num);
        }

        sc.close();
    }
}

/*
-----------------------------------------------
 VS CODE TERMINAL COMMANDS
```

---

javac slip16a.java

java slip16a

SAMPLE OUTPUT

Enter a decimal number (or -1 to exit): 25

Hexadecimal: 19

Octal: 31

Binary: 11001

Enter a decimal number (or -1 to exit): 100

Hexadecimal: 64

Octal: 144

Binary: 1100100

Enter a decimal number (or -1 to exit): -1

Exiting...

---

*/

Slip 16b

Q2 Write a program to connect with the available Wi-Fi using Arduino.

```
#include <SPI.h>
#include <WiFiNINA.h>

char ssid[] = "Wifi name";         // Your WiFi name
char pass[] = "secret password";   // Your WiFi password

int status = WL_IDLE_STATUS;       // WiFi status variable
void setup() {
  Serial.begin(9600);
  while (!Serial);                 // Wait for serial monitor

  // Try connecting to WiFi
  while (status != WL_CONNECTED) {
    Serial.print("Connecting to ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);   // Start WiFi connection
    delay(5000);                       // Wait 5 seconds
  }
  // When connected
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());      // Print local IP
}
void loop() {
  // Nothing required here
}
```

Slip 17

Slip 17a

Q1 Write a Java Program to implement Abstract Factory Pattern for Shape interface.

/*

Slip 17a

Q1 Write a Java Program to implement Abstract Factory Pattern for Shape interface.

*/

// Import must be at the top

import java.util.Scanner;

// Shape Interface

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

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

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

// Abstract Factory Interface
interface ShapeFactory {
    Shape createShape(String type);
}
```

```
// Concrete Factory
class ConcreteShapeFactory implements ShapeFactory {
    public Shape createShape(String type) {
        if(type.equalsIgnoreCase("CIRCLE")) return new Circle();
        else if(type.equalsIgnoreCase("RECTANGLE")) return new Rectangle();
        else return null;
    }
}

// Main Class
public class slip17a {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ShapeFactory factory = new ConcreteShapeFactory();

        System.out.print("Enter shape to draw (CIRCLE/RECTANGLE): ");
        String type = sc.nextLine();

        Shape shape = factory.createShape(type);
        if(shape != null) {
```

```
            shape.draw();
        } else {
            System.out.println("Invalid shape type");
        }

        sc.close();
    }
}

/*
-----------------------------------------------
 VS CODE TERMINAL COMMANDS
-----------------------------------------------
javac slip17a.java
java slip17a

-----------------------------------------------
 SAMPLE OUTPUT
-----------------------------------------------
Enter shape to draw (CIRCLE/RECTANGLE): CIRCLE
Drawing Circle
```

Enter shape to draw (CIRCLE/RECTANGLE): RECTANGLE

Drawing Rectangle

Enter shape to draw (CIRCLE/RECTANGLE): TRIANGLE

Invalid shape type

---

*/

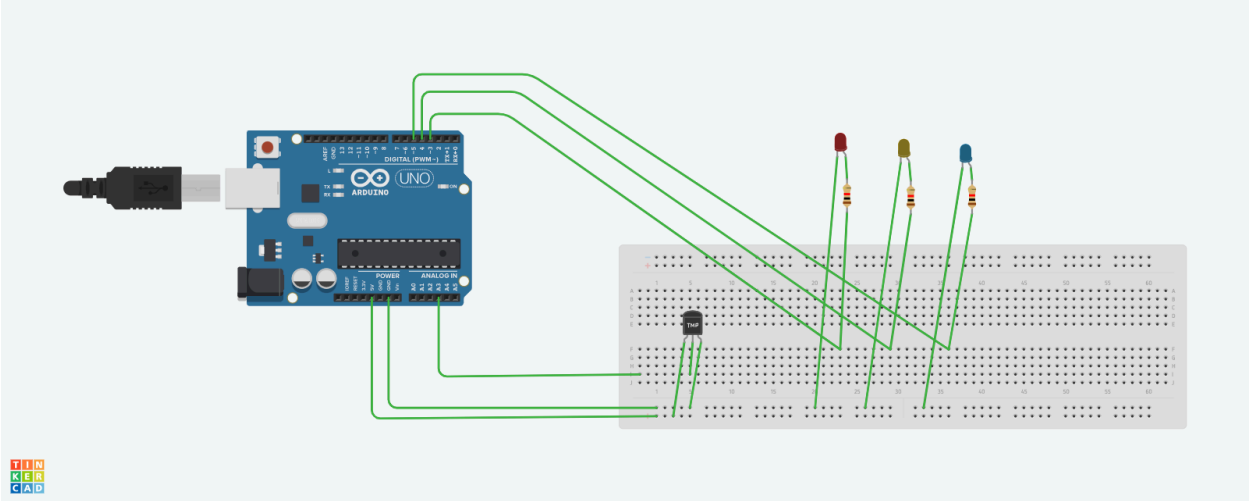
Slip 17b

Q2 Write a program to get temperature notification using Arduino.

```
const int hot = 90;     // Hot temperature threshold (F)
const int cold = 60;    // Cold temperature threshold (F)

// Pin definitions
// A2 = Temperature sensor input
// 2 = Red LED
// 3 = Blue LED
// 4 = Green LED

void setup() {
  pinMode(A2, INPUT);    // Sensor pin
  pinMode(2, OUTPUT);    // Red LED
  pinMode(3, OUTPUT);    // Blue LED
```

```
  pinMode(4, OUTPUT);    // Green LED

  Serial.begin(9600);    // Start serial monitor
}

void loop() {
  int sensor = analogRead(A2);                  // Read sensor value
  float voltage = (sensor / 1024.0) * 5.0;      // Convert to voltage
  float tempC = (voltage - 0.5) * 100;          // Convert to Celsius
  float tempF = (tempC * 1.8) + 32;             // Convert °C → °F

  Serial.print("temp: ");
  Serial.print(tempF);

  // Cold Condition
  if (tempF < cold) {
    digitalWrite(2, HIGH);   // Red ON
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    Serial.println(" It's Cold.");
  }
  // Hot Condition
  else if (tempF >= hot) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, HIGH);   // Green ON
    Serial.println(" It's Hot.");
  }
  // Fine Condition
  else {
    digitalWrite(2, LOW);
    digitalWrite(3, HIGH);   // Blue ON
    digitalWrite(4, LOW);
```

```
    Serial.println(" It's Fine.");
  }

  delay(400);
}
```



Arduino Uno/Genuino
Breadboard
3 resistors
3 RGB LEDs (any color)
Temperature Sensor [TMP36]

Slip 18

Slip 18a

Q1 Write a JAVA Program to implement built-in support (java.uti1.Observable) Weather station with members temperature, humidity, pressure and methods mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(),

getPressure()

/*

Slip 18a

Q1 Write a JAVA Program to implement Observer pattern for Weather Station.

Accept temperature, humidity, pressure and display updated values.

*/

```
import java.util.Scanner;

import java.util.List;

import java.util.ArrayList;


// Observer interface
```

```
interface Observer {
    void update(float temperature, float humidity, float pressure);
}

// Subject (Observable)
class WeatherData {
    private float temperature;
    private float humidity;
    private float pressure;
    private List<Observer> observers = new ArrayList<>();

    public void addObserver(Observer obs) {
        observers.add(obs);
    }

    public void removeObserver(Observer obs) {
        observers.remove(obs);
    }

    public void setMeasurements(float temperature, float humidity, float
pressure) {
```

```
        this.temperature = temperature;
        this.humidity = humidity;
        this.pressure = pressure;
        notifyObservers();
    }

    private void notifyObservers() {
        for(Observer obs : observers) {
            obs.update(temperature, humidity, pressure);
        }
    }
}

// Concrete Observer
class CurrentConditionsDisplay implements Observer {
    public void update(float temperature, float humidity, float pressure) {
        System.out.println("Current conditions: " + temperature + "°C, "
                           + humidity + "% humidity, " + pressure + " hPa");
    }
}
```

```
// Main class
public class slip18a {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        WeatherData weatherData = new WeatherData();

        CurrentConditionsDisplay display = new CurrentConditionsDisplay();
        weatherData.addObserver(display);

        while(true) {
            System.out.print("Enter temperature, humidity, pressure (-1 to exit): ");
            float temp = sc.nextFloat();
            if(temp == -1) break;
            float hum = sc.nextFloat();
            float pres = sc.nextFloat();
            weatherData.setMeasurements(temp, hum, pres);
        }

        System.out.println("Exiting...");
        sc.close();
    }
```

}

/*

VS CODE TERMINAL COMMANDS

---

javac slip18a.java

java slip18a

SAMPLE OUTPUT

Enter temperature, humidity, pressure (-1 to exit): 25 60 1013

Current conditions: 25.0°C, 60.0% humidity, 1013.0 hPa

Enter temperature, humidity, pressure (-1 to exit): 30 55 1010

Current conditions: 30.0°C, 55.0% humidity, 1010.0 hPa

Enter temperature, humidity, pressure (-1 to exit): -1

Exiting...

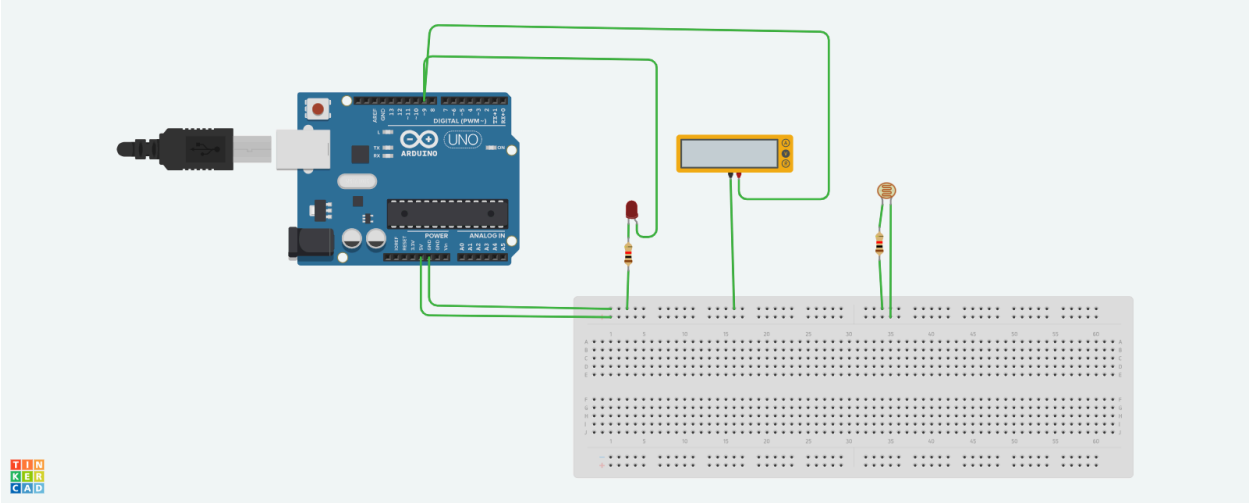
---

*/

Slip 18b

Q2 Write a program for LDR to vary the light intensity of LED using Arduino.

```
int sensorValue = 0;   // store LDR value

void setup() {
  pinMode(A0, INPUT);     // LDR input pin
  Serial.begin(9600);     // begin serial monitor
  pinMode(9, OUTPUT);     // LED on PWM pin 9
}

void loop() {
  sensorValue = analogRead(A0);  // Read LDR value (0–1023)

  Serial.println(sensorValue);   // Print for testing

  // Map LDR value to LED brightness (0–255)
  analogWrite(9, map(sensorValue, 0, 1023, 0, 255));

  delay(100);   // Small delay for stability
}
```



Arduino UNO
Photoresistor
Multimeter
LDR(Photoregister)
Resistors Breadboard LED

Slip 19

Slip 19a

Q1 Write a Java Program to implement Factory method for Pizza Store with createPizza(), orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizza's like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.

```
/*
Slip 19a
Q1 Java Program to implement Factory Method for Pizza Store.
Supports Delhi and Mumbai style Paneer Cheese pizzas.
*/


import java.util.Scanner;


// Abstract Pizza class
abstract class Pizza {
    String name;
    abstract void prepare();
    void bake() { System.out.println(name + " baking"); }
    void cut()  { System.out.println(name + " cutting"); }
    void box()  { System.out.println(name + " boxing"); }
}
```

```
// Concrete Pizzas
class DelhiPaneerPizza extends Pizza {
    DelhiPaneerPizza() { name = "Delhi Style Paneer Pizza"; }
    void prepare() { System.out.println(name + " preparing"); }
}

class MumbaiPaneerPizza extends Pizza {
    MumbaiPaneerPizza() { name = "Mumbai Style Paneer Pizza"; }
    void prepare() { System.out.println(name + " preparing"); }
}

// Factory
abstract class PizzaStore {
    abstract Pizza createPizza(String type);

    Pizza orderPizza(String type) {
        Pizza pizza = createPizza(type);
        if(pizza != null) {
            pizza.prepare();
            pizza.bake();
```

```
            pizza.cut();
            pizza.box();
        } else {
            System.out.println("Sorry, we don't have that pizza.");
        }
        return pizza;
    }
}

// Delhi Pizza Store
class DelhiStore extends PizzaStore {
    Pizza createPizza(String type) {
        if(type.equalsIgnoreCase("paneer")) return new DelhiPaneerPizza();
        return null;
    }
}

// Mumbai Pizza Store
class MumbaiStore extends PizzaStore {
    Pizza createPizza(String type) {
        if(type.equalsIgnoreCase("paneer")) return new MumbaiPaneerPizza();
```

```
        return null;
    }
}

// Main class
public class slip19a {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        PizzaStore delhi = new DelhiStore();
        PizzaStore mumbai = new MumbaiStore();

        System.out.print("Enter pizza type (paneer): ");
        String type = sc.next();

        System.out.print("Enter store (Delhi/Mumbai): ");
        String store = sc.next();

        if(store.equalsIgnoreCase("Delhi")) {
            delhi.orderPizza(type);
        } else if(store.equalsIgnoreCase("Mumbai")) {
            mumbai.orderPizza(type);
```

```
        } else {
            System.out.println("Invalid store");
        }

        sc.close();
    }
}

/*
-----------------------------------------------
VS CODE TERMINAL COMMANDS
-----------------------------------------------
javac slip19a.java
java slip19a

-----------------------------------------------
SAMPLE INPUT/OUTPUT
-----------------------------------------------
Enter pizza type (paneer): paneer
Enter store (Delhi/Mumbai): Delhi
Delhi Style Paneer Pizza preparing
```

Delhi Style Paneer Pizza baking

Delhi Style Paneer Pizza cutting

Delhi Style Paneer Pizza boxing

Enter pizza type (paneer): paneer

Enter store (Delhi/Mumbai): Mumbai

Mumbai Style Paneer Pizza preparing

Mumbai Style Paneer Pizza baking

Mumbai Style Paneer Pizza cutting

Mumbai Style Paneer Pizza boxing

---

*/

Slip 19b

Q2 Start Raspberry Pi and execute various Linux commands in the command terminal window: **ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping** etc.

Here are the **main Linux commands** you must execute on the Raspberry Pi with their simple purpose:

- **ls** – List files and folders
- **cd** – Change directory
- **touch** – Create an empty file
- **mv** – Move or rename a file
- **rm** – Remove/delete a file
- **man** – Show manual/help for a command
- **mkdir** – Create a new folder
- **rmdir** – Remove an empty folder
- **tar** – Create/extract archive files
- **gzip** – Compress or decompress files
- **cat** – View contents of a file
- **more** – View long files page-by-page
- **less** – View files with scroll up/down
- **ps** – Show running processes
- **sudo** – Run command as administrator
- **cron** – Schedule automated tasks

- **chown** – Change file owner
- **chgrp** – Change file group
- **ping** – Test network connection

Slip 20

Slip 20a

Q1 Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.
/*

Slip 20a

Q1 Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.

```
*/

import java.io.*;

public class slip20a {
    public static void main(String[] args) {
        try {
            // Input from terminal
            System.out.print("Enter text: ");
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

            // Read input line
            String input = br.readLine();

            // Convert to lowercase using decorator (FilterReader style)
            StringReader sr = new StringReader(input);
            LowerCaseReader lcr = new LowerCaseReader(sr);

            int c;
            System.out.print("Output in lowercase: ");
```

```
            while((c = lcr.read()) != -1) {
                System.out.print((char)c);
            }
            System.out.println();

            lcr.close();
            br.close();
        } catch(IOException e) {
            System.out.println("Error: " + e);
        }
    }
}

// Custom Decorator Reader
class LowerCaseReader extends FilterReader {
    protected LowerCaseReader(Reader in) {
        super(in);
    }

    @Override
    public int read() throws IOException {
```

```
        int c = super.read();
        if(c == -1) return c;
        return Character.toLowerCase((char)c);
    }

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

/*
-----------------------------------------------
VS CODE TERMINAL COMMANDS
-----------------------------------------------
```

javac slip20a.java

java slip20a

SAMPLE INPUT/OUTPUT

Enter text: HeLLo WoRLd

Output in lowercase: hello world

---

*/

Slip 20b
Q2 Write python programs on Raspberry Pi to perform the following:

Read your name and print a “Hello” message with the name.
Read two numbers and print their sum, difference, product, and division.
Count the number of words and characters of a given string.
Find the area of a given shape (rectangle, triangle, and circle) by reading the shape and appropriate values from standard input.

----

1. Read name and print Hello message

```
name = input("Enter your name: ")
print("Hello", name + "! Welcome to Raspberry Pi.")

# ---------------------------------------
# 2. Read two numbers and print arithmetic results
# ---------------------------------------
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

print("Sum =", num1 + num2)
print("Difference =", num1 - num2)
print("Product =", num1 * num2)

if num2 != 0:
    print("Division =", num1 / num2)
else:
    print("Division not possible (cannot divide by zero)")

# ---------------------------------------
# 3. Word and character count in a string
# ---------------------------------------
```

```
text = input("Enter a sentence: ")

word_count = len(text.split())
char_count = len(text)

print("Word Count:", word_count)
print("Character Count:", char_count)

# ----------------------------------------
# 4. Area of a selected shape
# ----------------------------------------
shape = input("Enter shape (rectangle/triangle/circle): ").lower()

if shape == "rectangle":
    length = float(input("Enter length: "))
    width = float(input("Enter width: "))
    area = length * width
    print("Area of Rectangle =", area)

elif shape == "triangle":
    base = float(input("Enter base: "))
```

```
    height = float(input("Enter height: "))
    area = 0.5 * base * height
    print("Area of Triangle =", area)

elif shape == "circle":
    radius = float(input("Enter radius: "))
    area = 3.14159 * radius * radius
    print("Area of Circle =", area)

else:
    print("Invalid shape entered!")
```