```
/*
  Sensor-Based Parking Assistance System
  Arduino Uno Code
  Includes:
  - 4 Ultrasonic Sensors (Front, Back, Left, Right)
  - Buzzer alerts
  - LEDs (Front, Back, Side indicators)
  - Bluetooth HC-05 (data sent to mobile app)
*/

// === Sensor Pin Definitions ===
const int trigFront = 2, echoFront = 3;
const int trigBack  = 4, echoBack  = 5;
const int trigLeft  = 6, echoLeft  = 7;
const int trigRight = 8, echoRight = 9;

// === Output Devices ===
const int buzzerPin = 10;
const int ledFront = 11;
const int ledBack  = 12;
const int ledSide  = 13;

// === Function to measure distance from a sensor ===
int getDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
  long duration = pulseIn(echoPin, HIGH, 30000); // timeout at 30ms (~5m)
  if (duration == 0) return 999; // No echo (too far or no object)

  int distance = duration * 0.034 / 2; // cm
  return distance;
}

void setup() {
  Serial.begin(9600);  // For HC-05 Bluetooth communication

  // Sensor pins
  pinMode(trigFront, OUTPUT); pinMode(echoFront, INPUT);
  pinMode(trigBack,  OUTPUT); pinMode(echoBack, INPUT);
  pinMode(trigLeft,  OUTPUT); pinMode(echoLeft, INPUT);
  pinMode(trigRight, OUTPUT); pinMode(echoRight, INPUT);

  // Outputs
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledFront, OUTPUT);
  pinMode(ledBack,  OUTPUT);
  pinMode(ledSide,  OUTPUT);

  Serial.println("✅ Parking Assist System Ready");
}

void loop() {
  // === Get distances ===
  int distFront = getDistance(trigFront, echoFront);
```

```
  int distBack  = getDistance(trigBack, echoBack);
  int distLeft  = getDistance(trigLeft, echoLeft);
  int distRight = getDistance(trigRight, echoRight);

  // === Send to Bluetooth (mobile app will receive this text) ===
  Serial.print("Front:"); Serial.print(distFront); Serial.print("cm ");
  Serial.print("Back:");  Serial.print(distBack);  Serial.print("cm ");
  Serial.print("Left:");  Serial.print(distLeft);  Serial.print("cm ");
  Serial.print("Right:"); Serial.print(distRight); Serial.println("cm");

  // === Reset outputs ===
  digitalWrite(buzzerPin, LOW);
  digitalWrite(ledFront, LOW);
  digitalWrite(ledBack, LOW);
  digitalWrite(ledSide, LOW);

  // === Process obstacle warnings ===
  processSensor(distFront, ledFront, "Front");
  processSensor(distBack,  ledBack,  "Back");
  processSensor(distLeft,  ledSide,  "Left");
  processSensor(distRight, ledSide,  "Right");

  delay(200); // Stabilize readings
}

// === Function to handle warnings ===
void processSensor(int distance, int ledPin, String direction) {
  if (distance > 0 && distance <= 15) {
    // 🚨 Very close → continuous alarm
```

```
    digitalWrite(buzzerPin, HIGH);
    digitalWrite(ledPin, HIGH);
    Serial.println("⚠ DANGER " + direction + "!");
  }
  else if (distance > 15 && distance <= 30) {
    // ⚠ Medium → fast beeps
    digitalWrite(ledPin, HIGH);
    digitalWrite(buzzerPin, HIGH);
    delay(100);
    digitalWrite(buzzerPin, LOW);
    delay(100);
  }
  else if (distance > 30 && distance <= 60) {
    // 🙂 Safe range → slow beeps
    digitalWrite(ledPin, HIGH);
    digitalWrite(buzzerPin, HIGH);
    delay(300);
    digitalWrite(buzzerPin, LOW);
    delay(300);
  }
}
```