Here's a detailed breakdown of all the topics for your **PL/SQL and DBMS Lab Internal Exam**, with explanations and sample code for each. I'll cover both logic-based programs (like pattern printing) and PL/SQL features (procedures, functions, triggers, etc.).

🔸 1. **Pattern Printing (Using , 1, 2, 3...)*

a) Star Pattern (Pyramid)

```
DECLARE
 i NUMBER;
 j NUMBER;
BEGIN
 FOR i IN 1..5 LOOP
  FOR j IN 1..i LOOP
   DBMS_OUTPUT.PUT('* ');
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('');
 END LOOP;
END;
```

b) Number Pattern (Triangle)

```
DECLARE
 i NUMBER;
 j NUMBER;
BEGIN
 FOR i IN 1..5 LOOP
  FOR j IN 1..i LOOP
   DBMS_OUTPUT.PUT(j || ' ');
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('');
 END LOOP;
END;
```

🔸 2. Palindrome Check

A number is a palindrome if it reads the same backward (e.g., 121)

```
DECLARE
 n NUMBER := 121;
 rev NUMBER := 0;
 rem NUMBER;
 temp NUMBER;
BEGIN
 temp := n;
 WHILE n > 0 LOOP
  rem := MOD(n, 10);
  rev := rev * 10 + rem;
  n := TRUNC(n / 10);
 END LOOP;
 IF temp = rev THEN
  DBMS_OUTPUT.PUT_LINE('Palindrome');
 ELSE
  DBMS_OUTPUT.PUT_LINE('Not Palindrome');
 END IF;
END;
```

- **3. Armstrong Number**

A number is Armstrong if the sum of cubes of digits equals the number (e.g., 153 = $1^3 + 5^3 + 3^3$)

```
DECLARE
 n NUMBER := 153;
 sum NUMBER := 0;
 rem NUMBER;
 temp NUMBER;
BEGIN
 temp := n;
 WHILE n > 0 LOOP
  rem := MOD(n, 10);
```

```
  sum := sum + rem * rem * rem;
  n := TRUNC(n / 10);
 END LOOP;
 IF temp = sum THEN
  DBMS_OUTPUT.PUT_LINE('Armstrong Number');
 ELSE
  DBMS_OUTPUT.PUT_LINE('Not Armstrong');
 END IF;
END;
```

🔸 **4. Fibonacci Series**

```
DECLARE
 a NUMBER := 0;
 b NUMBER := 1;
 c NUMBER;
 i NUMBER;
BEGIN
 DBMS_OUTPUT.PUT_LINE(a);
 DBMS_OUTPUT.PUT_LINE(b);
 FOR i IN 1..8 LOOP
  c := a + b;
  DBMS_OUTPUT.PUT_LINE(c);
  a := b;
  b := c;
 END LOOP;
END;
```

🔸 **5. Factorial of a Number**

```
DECLARE
 n NUMBER := 5;
 fact NUMBER := 1;
```

```
 i NUMBER;
BEGIN
 FOR i IN 1..n LOOP
  fact := fact * i;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Factorial is ' || fact);
END;
```

🔸 6. Arrays in PL/SQL (Using Collections)

a) Creating and Printing Array

```
DECLARE
 TYPE arr_type IS VARRAY(5) OF NUMBER;
 arr arr_type := arr_type(10, 20, 30, 40, 50);
 i NUMBER;
BEGIN
 FOR i IN 1..arr.COUNT LOOP
  DBMS_OUTPUT.PUT_LINE('Element ' || i || ' = ' || arr(i));
 END LOOP;
END;
```

🔸 7. Procedures

Procedures perform tasks but do not return a value.

```
CREATE OR REPLACE PROCEDURE greet_user(name IN VARCHAR2) IS
BEGIN
 DBMS_OUTPUT.PUT_LINE('Hello, ' || name);
END;
/

BEGIN
 greet_user('Prasanth');
END;
```

🔸 8. Functions

Functions return a value.

```
CREATE OR REPLACE FUNCTION square(x IN NUMBER)
RETURN NUMBER IS
BEGIN
 RETURN x * x;
END;
/

DECLARE
 res NUMBER;
BEGIN
 res := square(6);
 DBMS_OUTPUT.PUT_LINE('Square is: ' || res);
END;
```

🔸 9. Packages

Group procedures, functions, variables, etc.

a) Package Specification

```
CREATE OR REPLACE PACKAGE math_pkg IS
 FUNCTION add(a NUMBER, b NUMBER) RETURN NUMBER;
 PROCEDURE say_hello;
END;
/
```

b) Package Body

```
CREATE OR REPLACE PACKAGE BODY math_pkg IS
 FUNCTION add(a NUMBER, b NUMBER) RETURN NUMBER IS
 BEGIN
  RETURN a + b;
 END;
```

```
 PROCEDURE say_hello IS
 BEGIN
  DBMS_OUTPUT.PUT_LINE('Hello from Package');
 END;
END;
/
```

c) Using Package

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('Sum: ' || math_pkg.add(10, 20));
 math_pkg.say_hello;
END;
```

🔸 10. Exception Handling

a) Predefined Exception

```
BEGIN
 DECLARE
  a NUMBER := 10;
  b NUMBER := 0;
  c NUMBER;
 BEGIN
  c := a / b;
  DBMS_OUTPUT.PUT_LINE('Result: ' || c);
 EXCEPTION
  WHEN ZERO_DIVIDE THEN
   DBMS_OUTPUT.PUT_LINE('Cannot divide by zero.');
 END;
END;
```

b) User-Defined Exception

```
DECLARE
 age NUMBER := 15;
```

```
 underage EXCEPTION;
BEGIN
 IF age < 18 THEN
  RAISE underage;
 END IF;
EXCEPTION
 WHEN underage THEN
  DBMS_OUTPUT.PUT_LINE('Underage not allowed.');
END;
```

🔸 11. Cursors

a) Implicit Cursor

```
BEGIN
 INSERT INTO students VALUES (101, 'Prasanth');
 IF SQL%ROWCOUNT > 0 THEN
  DBMS_OUTPUT.PUT_LINE('1 row inserted');
 END IF;
END;
```

b) Explicit Cursor

```
DECLARE
 CURSOR cur IS SELECT name FROM students;
 student_name students.name%TYPE;
BEGIN
 OPEN cur;
 LOOP
  FETCH cur INTO student_name;
  EXIT WHEN cur%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Student: ' || student_name);
 END LOOP;
 CLOSE cur;
END;
```

🔸 **12. Triggers**

a) BEFORE INSERT Trigger

```
CREATE OR REPLACE TRIGGER trg_before_insert
BEFORE INSERT ON students
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('About to insert a new student');
END;
/
```

Let me know if you want **PDF notes**, **practice questions**, or any **specific topic elaboration**.