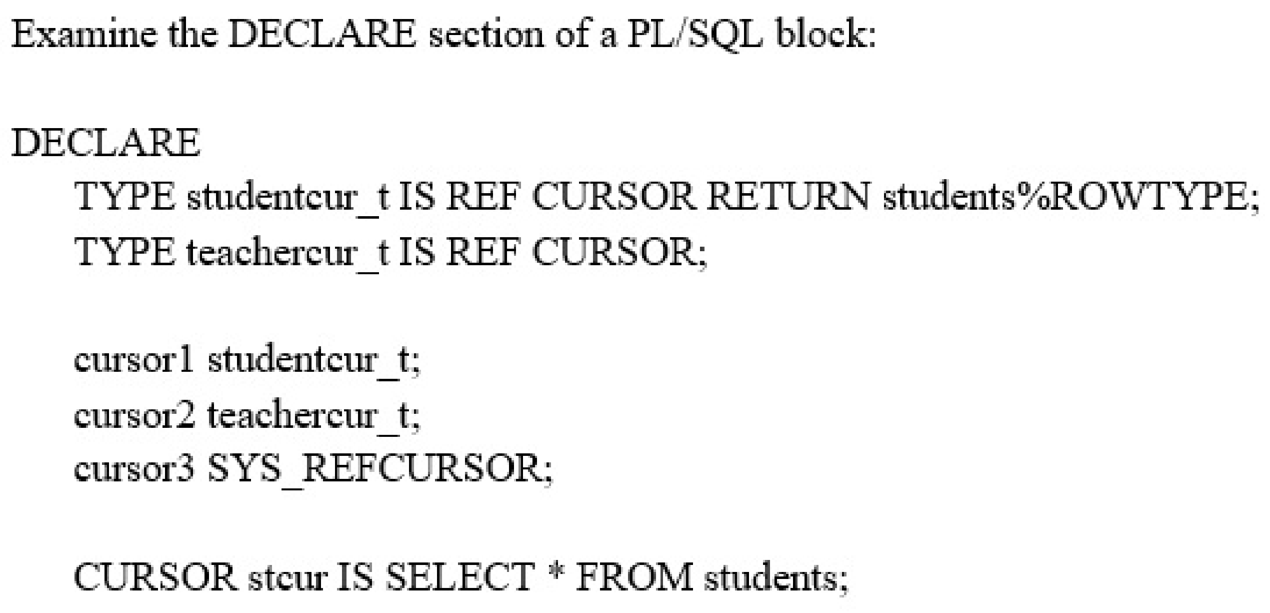
Version: 8.0

Question: 1

The STUDENTS table exists in your schema.
Examine the DECLARE section of a PL/SQL block:



Examine the DECLARE section of a PL/SQL block:

```
DECLARE
    TYPE studentcur_t IS REF CURSOR RETURN students%ROWTYPE;
    TYPE teachercur_t IS REF CURSOR;

    cursor1 studentcur_t;
    cursor2 teachercur_t;
    cursor3 SYS_REFCURSOR;

    CURSOR stcur IS SELECT * FROM students;
```

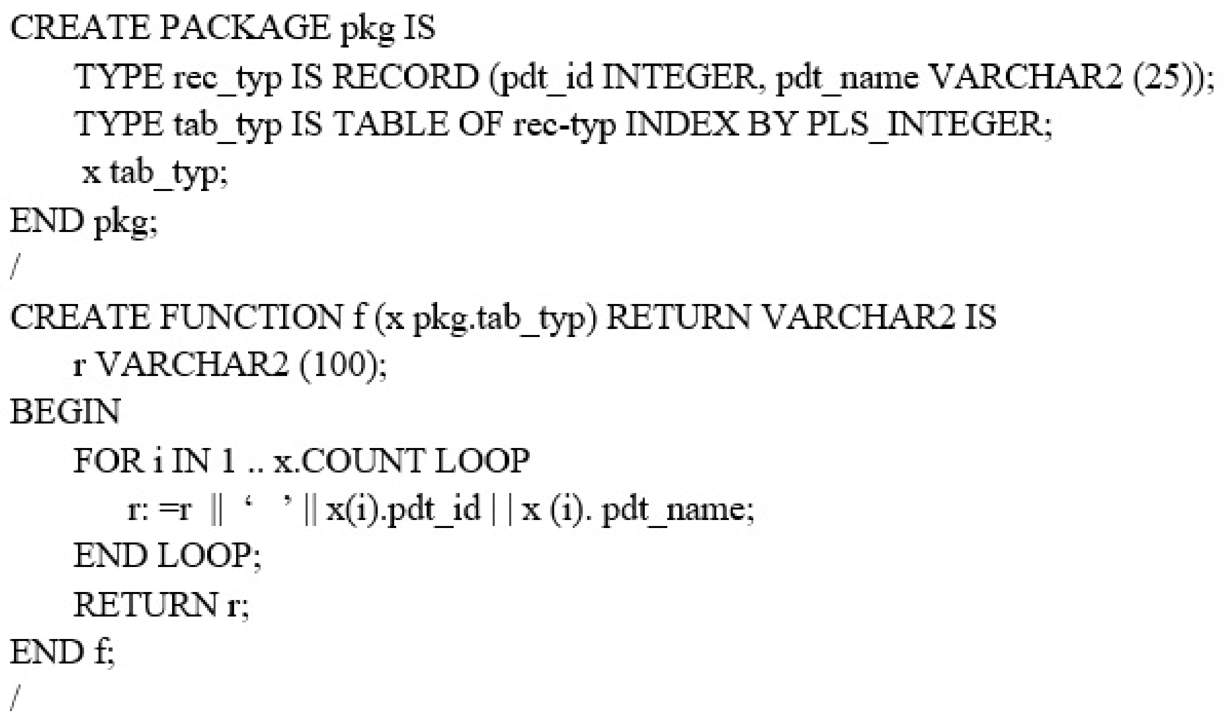
Which two blocks are valid?

A. BEGINOPEN cursor3 FOR SELECT * FROM students;cursor1 :=cursor3;END;
B. BEGINOPEN stcur;cursor1 :=stcur;END;
C. BEGINOPEN cursor1 FOR SELECT * FROM students;stcur :=cursor1;END;
D. BEGINOPEN stcur;cursor3 :=stcur;END;
E. BEGINOPEN cursor1 FOR SELECT * FROM students;cursor2 :=cursor1;END;

Answer: D,E

Question: 2

Examine the code:



```
CREATE PACKAGE pkg IS
     TYPE rec_typ IS RECORD (pdt_id INTEGER, pdt_name VARCHAR2 (25));
     TYPE tab_typ IS TABLE OF rec-typ INDEX BY PLS_INTEGER;
      x tab_typ;
END pkg;
/
CREATE FUNCTION f (x pkg.tab_typ) RETURN VARCHAR2 IS
     r VARCHAR2 (100);
BEGIN
     FOR i IN 1 .. x.COUNT LOOP
          r: =r  ||  '   ' || x(i).pdt_id | | x (i). pdt_name;
     END LOOP;
     RETURN r;
END f;
/
```

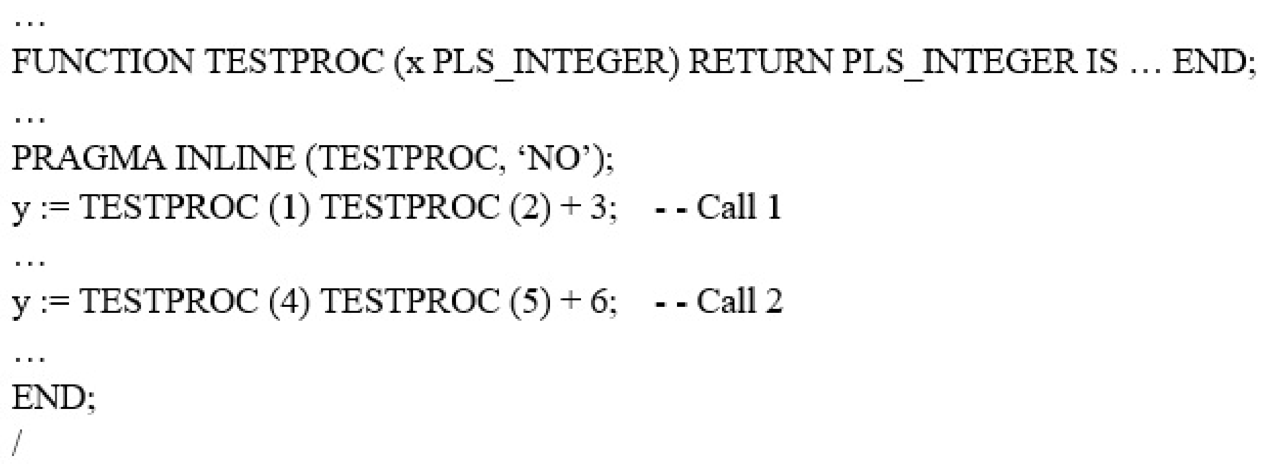
Which two subprograms will be created successfully?

A. CREATE FUNCTION p4 (y pkg.tab_typ) RETURN pkg.tab_typ ISBEGINEXECUTE IMMEDIATE 'SELECT pdt_id, pdt_name FROM TABLE (:b)'BULT COLLECT INTO pkg.x USING y;RETURN pkg.x;END p4;
B. CREATE PROCEDURE p1 (y IN OUT pkg.tab_typ) ISBEGINEXECUTE IMMEDIATE 'SELECT f (:b) FROM DUAL' INTO y USING pkg.x;END p1;
C. CREATE PROCEDURE p2 (v IN OUT VARCHAR2) ISBEGINEXECUTE IMMEDIATE 'SELECT f (:b) FROM DUAL' INTO v USING pkg.x;END p2;
D. CREATE FUNCTION p3 RETURN pkg. tab_typ ISBEGINEXECUTE IMMEDIATE 'SELECT f (:b) FROM DUAL' INTO pkg.x;END p3;
E. CREATE PROCEDURE p5 (y pkg. rec_typ) ISBEGINEXECUTE IMMEDIATE 'SELECT pdt_name FROM TABLE (:b)' BULK COLLECT INTO y USING pkg.x;END p5;

Answer: A,C

Question: 3

Examine the section of code taken from a PL/SQL program:



```
…
FUNCTION TESTPROC (x PLS_INTEGER) RETURN PLS_INTEGER IS … END;
…
PRAGMA INLINE (TESTPROC, ‘NO’);
y := TESTPROC (1) TESTPROC (2) + 3;    - - Call 1
…
y := TESTPROC (4) TESTPROC (5) + 6;    - - Call 2
…
END;
/
```

PLSQL_OPTIMIZE_LEVEL PARAMETER is set to 3.
Which two statements are true?

A. Calls to TESTPROC will always be inlined as it is compiled with PLSQL_OPTIMIZE_LEVEL=3.
B. Calls to TESTPROC are never inlined in both lines commented as Call1 and Call 2.
C. Calls to TESTPROC are not inlined in the line commented as Call 1.
D. Calls to TESTPROC are inlined in both lines commented as Call 1 and Call 2.
E. Calls to TESTPROC might be inlined in the line commented as Call 2.

Answer: A,E

Question: 4

Which statement is true about the DBMS_PARALLEL_EXECUTE package?

A. DBMS_PARALLEL_EXECUTE is a SYS-owned package and can be accessed only by a user with DBA privileges.
B. To execute chunks in parallel, users must have CREATE JOB system privilege.
C. No specific system privileges are required to create or run parallel execution tasks.
D. Only DBAs can create or run parallel execution tasks.
E. Users with CREATE TASK privilege can create or run parallel execution tasks.

Answer: B

Reference:
https://docs.oracle.com/cd/E11882_01/appdev.112/e40758/d_parallel_ex.htm#ARPLS67331(securit
y model)

Question: 5

Which two statements are true regarding edition-based redefinition (EBR)?

A. There is no default edition defined in the database.
B. EBR does not let you upgrade the database components of an application while in use.

C. You never use EBR to copy the database objects and redefine the copied objects in isolation.
D. Editions are non-schema objects.
E. When you change an editioned object, all of its dependents remain valid.
F. Tables are not editionable objects.

Answer: E,F

Question: 6

Which two blocks of code execute successfully?

A. DECLARESUBTYPE new_one IS BINARY_INTERGER RANGE 0..9;my_val new_one;BEGINmy_val :=0;END;
B. DECLARESUBTYPE new_string IS VARCHAR2 (5) NOT NULL;my_str_new_string;BEGINmy_str := ‘abc’;END;
C. DECLARESUBTYPE new_one IS NUMBER (2, 1);my_val new_one;BEGINmy_val :=12.5;END;
D. DECLARESUBTYPE new_one IS INTEGER RANGE 1..10 NOT NULL;my_val new_one;BEGINmy_val :=2;END;
E. DECLARESUBTYPE new_one IS NUMBER (1, 0);my_val new_one;BEGINmy_val := -1;END;

Answer: A,D

Question: 7

Which statement is correct about DBMS_LOB.SETOPTIONS and DBMS_LOB.GETOPTIONS for SecureFiles?

A. DBMS_LOB.GETOPTIONS can only be used for BLOB data types.
B. DBMS_LOB.SETOPTIONS can perform operations on individual SecureFiles but not an entire column.
C. DBMS_LOB. SETOPTIONS can set option types COMPRESS, DUPLICATE, and ENCRYPT.
D. If a table was not created with compression specified in the store as securefile clause then DBMS_LOB.SETOPTIONS can be used to enable it later.

Answer: D

Question: 8

You are designing and developing a complex database application built using many dynamic SQL statements. Which option could expose your code to SQL injection attacks?

A. Using bind variables instead of directly concatenating parameters into dynamic SQL statements
B. Using automated tools to generate code
C. Not validating parameters which are concatenated into dynamic SQL statements
D. Validating parameters before concatenating them into dynamic SQL statements
E. Having excess database privileges

Answer: A

Question: 9

Examine this code executed as SYS:

```
CREATE USER spider IDETIFIED BY spider DEFAULT TABLESPACE users QUOTA
UNLIMITED ON users;
CREATE ROLE dynamic_table_role;
GRANT CREATE TABLE TO dynamic_table_role;
GRANT CREATE SESSION, CREATE PROCEDURE TO spider;
GRANT dynamic_table_role TO spider WITH ADMIN OPTION;
ALTER USER spider DEFAULT ROLE ALL EXCEPT dynamic_table_role;
```

Examine this code executed as SPIDER and the error message received upon execution:

```
CREATE PROCEDURE dproc AS
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE demo (id INTEGER)';
END;
/
SET ROLE dynamic_table_role;
EXEC dproc;

ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SPIDER.DPROC", line 4
ORA-06512: at line 1
```

What is the reason for this error?

A. The procedure needs to be granted the DYNAMIC_TABLE_ROLE role.
B. The EXECUTE IMMEDIATE clause is not supported with roles.
C. Privileges granted through roles are never in effect when running definer's rights procedures.
D. The user SPIDER needs to be granted the CREATE TABLE privilege and the procedure needs to be granted the DYNAMIC_TABLE_ROLE.

Answer: C

Question: 10

Which codes executes successfully?

A. CREATE PACKAGE pkg ASTYPE rec_typ IS RECORD (price NUMBER, inc_pct NUMBER);PROCEDURE calc_price (price_rec IN OUT rec_typ);END pkg;/CREATE PACAKGE BODY pkg ASPROCEDURE calc_price (price_rec IN OUT rec_typ) ASBEGINprice_rec.price := price_rec.price + (price_rec.price * price_rec.inc_pct)/100;END calc_price;END pkg;/DECLARE1_rec pkg. rec_typ;BEGIN1_rec_price :=100;1_rec.inc_pct :=50;EXECUTE IMMEDIATE ‘BEGIN pkg. calc_price (:rec); END;’ USING IN OUT 1_rec;END;
B. CREATE PACKAGE pkg ASTYPE rec_typ IS RECORD (price NUMBER, inc_pct NUMBER);END pkg;/CREATE PROCEDURE calc_price (price_rec IN OUT pkg. rec_typ) ASBEGINprice_rec.price := price_rec.price + (price_rec.price * price_rec.inc_pct)/100;END/DECLARE1_rec pkg.rec_typ;BEGINEXECUTE IMMEDIATE ‘BEGIN calc_price (:rec); END;’ USING IN OUT 1_rec (100, 50);END;
C. CREATE PACKAGE pkg ASTYPE rec_typ IS RECORD (price NUMBER, inc_pct NUMBER);END pkg;/CREATE PROCEDURE calc_price (price_rec IN OUT pkg. rec_typ) ASBEGINprice_rec.price := price_rec.price + (price_rec.price * price_rec.inc_pct)/100;END ;/DECLARE1_rec pkg. rec_typ;BEGIN1_rec_price :=100;1_rec.inc_pct :=50;EXECUTE IMMEDIATE ‘BEGIN calc_price (1_rec); END;’;END;
D. DECLARETYPE rec_typ IS RECORD (price NUMBER, inc_pct NUMBER);1_rec rec-typ;PROCEDURE calc_price (price_rec IN OUT rec_typ) ASBEGINprice_rec.price := price-rec.price+ (price_rec.price * price_rec.inc_pct)/100;END;BEGIN1_rec_price :=100;1_rec.inc_pct :=50;EXECUTE IMMEDIATE ‘BEGIN calc_price (:rec); END;’ USING IN OUT 1_rec;END;

Answer: B