1. Add a procedure to the school_api package called remove_student. This procedure accepts anstudent_id and returns nothing. Based on the student ID passed in, it removes the student from the database. If the student does not exist or if a problem occurs while removing the student (such as a foreign key constraint violation), let the calling program handle it.

SOLUTION:

```
/*package specification*/
CREATE OR REPLACE PACKAGE school_api AS
   PROCEDURE remove_student(p_student_id IN student.student_id%TYPE);
END school_api;

CREATE SEQUENCE INSTRUCTOR_ID_SEQ increment by 10;

/*package body*/
CREATE OR REPLACE PACKAGE BODY school_api AS
   PROCEDURE remove_student(p_student_id IN student.student_id%TYPE) IS
   BEGIN
         DELETE FROM student WHERE student_id = p_student_id;
   END;
END school_api;

SET SERVEROUTPUT ON
DECLARE
   v_student_id student.student_id%TYPE := &sv_student_id;
BEGIN
   school_api.remove_student(v_student_id);
END;
```

OUTPUT:

If the id exist in the table:

```
Error report -
ORA-02292: integrity constraint (SYS.ENR_STU_FK) violated - child record found
ORA-06512: at "SYS.SCHOOL_API", line 4
ORA-06512: at line 4
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:    attempted to delete a parent key value that had a foreign
           dependency.
*Action:   delete dependencies first then parent or disable constraint.
```

Otherwise:

```
PL/SQL procedure successfully completed.
```

2. Alter remove_student in the school_api package body to accept an additional parameter. This new parameter should be a VARCHAR2 and should be called p_ri. Make p_ri default to R. The new parameter may contain a value of R or C. If R is received, it represents DELETE RESTRICT, and the procedure acts as it does now. If there are enrollments for the student, the delete is disallowed. If a C is received, it represents DELETE CASCADE. This functionally means that the remove_student procedure locates all records for the student in all the Student Database tables. It removes them from the database before attempting to remove the student from the student table. Decide how to handle the situation when the user passes in a code other than C or R.

SOLUTION:

```
/*package specification*/
CREATE OR REPLACE PACKAGE school_api AS
   PROCEDURE remove_student(p_student_id IN student.student_id%TYPE,
                            p_ri VARCHAR2 DEFAULT 'C');
END school_api;

/*package body*/
CREATE OR REPLACE PACKAGE BODY school_api AS
   PROCEDURE remove_student(p_student_id IN student.student_id%TYPE,
                            p_ri VARCHAR2 DEFAULT 'C') IS
   student_exists EXCEPTION;
   not_valid_p_ri EXCEPTION;
   BEGIN
      IF p_ri = 'R' THEN
         DECLARE
         test_var CHAR(1);
         BEGIN
            SELECT NULL
            INTO test_var
            FROM enrollment e
            WHERE e.student_id = p_student_id AND ROWNUM = 1;
            RAISE student_exists;

            EXCEPTION
               WHEN NO_DATA_FOUND THEN
            DELETE FROM student WHERE student_id = p_student_id;
         END;
      ELSIF p_ri = 'C' THEN
         DELETE FROM enrollment WHERE student_id = p_student_id;
         DELETE FROM grade WHERE student_id = p_student_id;
         DELETE FROM student WHERE student_id = p_student_id;
      ELSE
```

```
            RAISE not_valid_p_ri;
        END IF;

        EXCEPTION
            WHEN not_valid_p_ri THEN
                DBMS_OUTPUT.PUT_LINE('Not a valid p_ri! Error!');
            WHEN student_exists THEN
                DBMS_OUTPUT.PUT_LINE('Student exist in other tables! Error!');
    END;
END school_api;

SET SERVEROUTPUT ON
DECLARE
    v_student_id student.student_id%TYPE := &sv_student_id;
BEGIN
    school_api.remove_student(v_student_id);
END;
```

OUTPUT: