1. Use a sequence to generate the values for logid automatically when new log records are inserted into the logs table. Start the sequence with 1000.

Usage - To generate log id
Objective - Automatic generation of primary key of Log table

```
-- Sequence for log table
CREATE SEQUENCE seq_logs
  INCREMENT BY 1
  START WITH 1000;
```

2. Write procedures in your package to display the tuples in each of the eight tables: students, courses, course_credit, prerequisites, classes, enrollments, grades, logs. As an example, you can write a procedure, say show_students, to display all students in the students table.

Usage - This procedure simply displays content of eight tables.
Objective – Check content of tables after execution of other procedure

```
PROCEDURE proc_show_all_tables(
        in_table_name VARCHAR2,
        out_prc OUT sys_refcursor );
```

3. Write a procedure in your package that, for a given student (with B# provided as a parameter), can list every class the student has taken or is taking (list all attributes except limit and class_size) as well as the letter grade and number grade the student received for the class (including null grades). If the classid is not in the Students table, report “The B# is invalid.” If the student has not taken any course, report “The student has not taken any course.”

Usage – To display student information
Objective - to check which class student has taken as well as letter grade , number grade the student received for this class. (including null grades). Report respective errors if applicatble.

```
PROCEDURE proc_show_student_info(
        in_student_B# IN students.B#%type,
        out_prc OUT sys_refcursor ) ;
```

4. Write a procedure in your package that, for a given course (with dept_code and course# as parameters), can return all courses that need this course as a prerequisite (show dept_code and course# together as in CS532), including courses that need the course as prerequisite either directly and indirectly. If course C2 has course C1 as a prerequisite, C1 is a direct prerequisite of C2. If C3 has course C2 has a prerequisite, then C1 is an indirect prerequisite for C3. Please note that indirect prerequisites can be more than two levels away

Usage – This procedure gives prerequisite of given course (direct and indirect)

Objective – to get direct/indirect prerequisite of given course, this may be useful while deleting enrollments or enrolling student in course.

```
PROCEDURE proc_find_dependent_courses(
      c_dept_code IN PREREQUISITES.pre_dept_code%TYPE,
      c_course#   IN PREREQUISITES.pre_course#%TYPE,
      out_prc OUT sys_refcursor );
```

5. (3 points) Write a procedure in your package that, for a given class (with classid provided as a parameter), can list the classid and course title of the class as well as all the students (show B# and firstname) who took or are taking the class. If the class is not in the classes table, report "The classid is invalid." If no student took or is taking the class, report "No student has enrolled in the class."

Usage – Give Class details of given class id
Objective – To get the student enrolled in a particulate class with class information. Report errors if applicable.

```
procedure proc_show_class_details (p_class_id in Classes.classid%type,
   out_prc OUT sys_refcursor);
```

6. Write a procedure in your package to enroll a student into a class. The B# of the student and the classid of the class are provided as parameters. If the student is not in the students table, report "The B# is invalid." If the class is not in the classes table, report "The classid is invalid." If the enrollment of the student into a class would cause "class_size > limit", reject the enrollment and report "The class is full." If the student is already in the class, report "The student is already in the class." If the student is already enrolled in three other classes in the same semester and the same year, report "You are overloaded." and allow the student to be enrolled. If the student is already enrolled in four other classes in the same semester and the same year, report "Students cannot be enrolled in more than four classes in the same semester." and reject the enrollment. If the student has not completed the required prerequisite courses with at least a grade C, reject the enrollment and report "Prerequisite not satisfied." For all the other cases, the requested enrollment should be carried out successfully. You should make sure that all data are consistent after each enrollment. For example, after you successfully enrolled a student into a class, the class size of the class should be updated accordingly. Use trigger(s) to implement the updates of values caused by successfully enrolling a student into a class. It is recommended that all triggers for this project be implemented outside of the package.

Usage - This procedure is useful to enroll student in a class.
Objective – Enroll given student in a given class. Report errors if applicable.

```
   PROCEDURE enroll_student(
```

```
stud_B#      IN STUDENTS.B#%TYPE,
stud_classid IN CLASSES.CLASSID%TYPE);
```

7.Write a procedure in your package to drop a student from a class (i.e., delete a tuple from the enrollments table). The B# of the student and the classid of the class are provided as parameters. If the student is not in the students table, report “The B# is invalid.” If the classid is not in the classes table, report “The classid is invalid.” If the student is not enrolled in the class, report “The student is not enrolled in the class.” If dropping the student from the class would cause a violation of the prerequisite requirement for another class, then reject the drop attempt and report “The drop is not permitted because another class uses it as a prerequisite.” In all the other cases, the student should be dropped from the class. If the class is the last class for the student, report “This student is
not enrolled in any classes.” If the student is the last student in the class, report “The class now has no students.” Again, you should make sure that all data are consistent after the drop and all updates caused by the drop should be implemented using trigger(s).

Usage – This procedure is useful for deleting enrollment of a student from enrollments table
Objective - Drop Student from enrollments

```
PROCEDURE PROC_DELETE_ENROLLMENT(
   p_B#      IN ENROLLMENTS.B#%TYPE,
   p_classid IN ENROLLMENTS.CLASSID%TYPE);
```

8. (5 points) Write a procedure in your package to delete a student from the students table based on a given B# (as a parameter). If the student is not in the students table, report “The B# is invalid.” When a student is deleted, all tuples in the enrollments table involving the student should also be deleted (use a trigger to implement this) and this will trigger a number of actions as described in the above item (item 7).

Usage – This procedure is useful for deleting a student from student table and report errors applicable.
Objective - Delete Student from Students table

```
PROCEDURE PROC_DELETE_STUDENT (p_students_b# IN STUDENTS.B#%TYPE);
```

9. (8 points) Write triggers to add tuples to the logs table automatically whenever a student is added to or deleted from the students table, or when a student is successfully enrolled into or dropped from a class (i.e., when a tuple is inserted into or deleted from the enrollments table). For a logs record for enrollments, the key value is the concatenation of the B# value, a comma, and the classid value.

Usage – Triggers are created wherever applicable In student registration system.

Objective – To log events in a database

Following Triggers created –

- trigger TRIGGER_DEC_CLASS_SIZE;
  - to decrement class size of class if any student is dropped from class
- trigger trigger_delete_enrollments;
  - to create a log entry when a enrollment deleted from enrollments table
- trigger trigger_insert_student;
  - to create a log entry when a student inserted
- trigger trigger_delete_student;
  - to create a log entry when a student deleted from student table
- trigger trigger_insert_enrollments;
  - to create a log entry when a enrollment inserted in enrollments table
- trigger trigger_inc_class_size;
  - to increment class size of class if any student is enrolled from class

few local functions created in package are as follows:

```
FUNCTION is_class_present(
     p_class_classid IN CLASSES.CLASSID%TYPE)
   RETURN INTEGER

FUNCTION is_student_present(
     p_student_B# IN Students.B#%type)
   RETURN INTEGER

FUNCTION is_student_enrolled(
     p_class_classid IN CLASSES.CLASSID%TYPE,
     p_student_B# Students.B#%type)
   RETURN INTEGER

FUNCTION is_any_student_enrolled(
     p_class_classid IN CLASSES.CLASSID%TYPE)
   RETURN INTEGER

FUNCTION GIVE_RESPECTIVE_NGRADE(
     IN_LGRADE IN VARCHAR2 )
   RETURN VARCHAR2

FUNCTION GIVE_RESPECTIVE_NGRADE
(
  IN_LGRADE IN VARCHAR2
) RETURN VARCHAR2;
```