**44-560 Adv Topics in DB Systems Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Exam 03 Part 2 (60 points) KEY** *please print*

***For each question, make your answer fit in the space provided. Do not write on the back of any page.***

***This exam uses the tables faculty, student, and student\_log described in a different document.***

1. (16 pts) Write a stored procedure named **display\_faculty** that displays the faculty id and name for each record in table **faculty**. Hint: Use a cursor.

**create or replace procedure display\_faculty is**

**facultyID faculty.facid%type;**

**facultyName faculty.facname%type;**

**cursor faculty\_cursor is**

**select facid, facname**

**from faculty;**

**begin**

**open faculty\_cursor;**

**loop**

**fetch faculty\_cursor into facultyID, facultyName;**

**exit when faculty\_cursor%notfound;**

**dbms\_output.put\_line(facultyID || ' ' || facultyName);**

**end loop;**

**close faculty\_cursor;**

**end;**

1. (4 pts) Write a single statement that you could use to execute the above procedure in SQL\*Plus or SQLDeveloper.

The statement you write must work at the SQL> prompt inside SQL\*Plus or in SQL Developer.

**execute display\_faculty; -- may replace execute with exec**

1. (4 pts) Write a single statement to create a sequence named **stuLogSeq** that begins with the value of 500.

**create sequence stuLogSeq start with 500;**

1. (12 pts) Write a row-level trigger named **stuGpa** that will insert appropriate information into the **student\_log** table each time the **student** table is updated. Use the sequence you created in the previous step to generate the value of **stuLogId**.

**create or replace trigger stuGpa**

**before update on student**

**for each row**

**begin**

**insert into student\_log values (stuLogSeq.nextval,**

**:OLD.stuid, :OLD.gpa, :NEW.gpa, user, sysdate);**

**end;**

1. (8 pts) Write a single SQL statement that lists each student’s name and the name of the student’s faculty advisor. Do not use the word **JOIN** in your statement.

For the sample data given, the result of your SQL statement should be as shown below. Note that the order of the rows is irrelevant.

**STUNAME FACNAME**

**-------------------- ------------**

**Hermione Brown**

**Harry Brown**

**Alex Brown**

**Susie Thompson**

**Andy Thompson**

**Betty Thompson**

**Imogene Thompson**

**select stuName, facName**

**from student, faculty**

**where student.facid = faculty.facid;**

1. (8 pts) Write a single SQL statement that lists each student’s name and the name of the student’s faculty advisor. A faculty member who has no advisees and a student who has no assigned advisor should also be included in the listing.

For the sample data given, the result of your SQL statement should be as shown below. Note that the order of the rows is irrelevant.

**STUNAME FACNAME**

**-------------------- ------------**

**Andy Thompson**

**Susie Thompson**

**Alex Brown**

**Betty Thompson**

**Hermione Brown**

**Harry Brown**

**Imogene Thompson**

**Becky**

**Anderson**

**select stuName, facName**

**from student full outer join faculty**

**on student.facid = faculty.facid;**

**Note: The word “outer” is optional; it can be omitted**

1. (8 pts) Write a single SQL statement that lists each student’s id, name, and gpa for those students whose gpa is greater than the average of all gpas.

For the sample data given, the result of your SQL statement should be as shown below. Note that the order of the rows is irrelevant.

**STUID STUNAME GPA**

**----- ------------ ---------**

**7777 Alex 4**

**6543 Hermione 4**

**1234 Betty 3.8**

**4321 Susie 3.9**

**5544 Becky 4**

**select stuid, stuname, gpa**

**from student**

**where gpa > (select avg(gpa) from student);**