CSC 352 / 452: Database Programming

assignment #2 (60 Points)

**CSC 352/452-501: Due on Tuesday, 7/28/2015 at 11:59PM**

**CSC 352/452-510: Due on Wednesday, 7/29/2015 at 11:59PM**

Unless prior arrangements are made, homework turned in late but within 24 hours of the due time will be graded at 75% credit, homework turned in between 24 and 48 hours will be graded at 50% credit, and homework turned in later than 48 hours will not be accepted.

**Please note that only TEXT files will be accepted. All other file types (e.g., DOC, DOCX, RTF, PDF, JPG, or ZIP) will be rejected. In D2L, only the most recent submission is kept.**

* If you modified the DEPARTMENT and EMPLOYEE tables created in Assignment #1, you need to delete and re-populate them.
* **Do not try to use ONE SELECT-INTO/JOIN statement to get the results.** You can use several SELECT-INTO statements to get the results easily.
* You cannot use hard-coded values (e.g., IF employee\_id = 7839 THEN ……) in your programs.
* You are not allowed to create temporary tables, views, functions, or procedures.
* Explicit cursors are NOT allowed in your programs.
* The EXCEPTION Section is NOT allowed in your programs.
* Please review your assignment file before submitting it to make sure you have the correct one. It is your responsibility to ensure that you upload the correct assignment file.

**1) (CSC 352 - 25 points | CSC 452 – 20 points)**

Based on the DEPARTMENT and EMPLOYEE tables created in Assignment #1, write a **PL/SQL anonymous block** that accepts an employee ID from the user input and displays the employee’s name, job, hire date, salary, commission, total pay (salary + commission), his/her manager’s name and hire date, and the **name** of the department where the employee works.

* Submitting more than **one** PL/SQL program will receive 0 points.
* If the employee ID from the user input is not in the EMPLOYEE table (EMPLOYEE\_ID), you display a message telling the user that the employee ID is not in the table.
* You must display salary/commission/total pay with a dollar ($) sign, a comma, and two decimal places (e.g., $1,234.56).
* The hire date must be displayed in the mm/dd/yyyy format (4-digit year) if it is not NULL.
* If the employee does not have a commission, the commission must be shown as $0.00 in your output.
* If the employee does not belong to any department, the department name must be shown as “\*\*\*\*\*\*” in your output.
* If the employee does not have a manager, the manager name and hire date must be shown as “\*\*\*\*\*\*” in your output.
* You will lose 5 points if your output does not have “=== Employee ===”, “=== Department ===”, or “=== Manager ===”.
* To avoid complicating issues, you can assume that the user always enters input from keyboard that consists only of the digits 0 through 9 and Enter.

Example:

ACCEPT p\_1 PROMPT 'Please enter the employee ID:'

DECLARE

-- *Your statements*

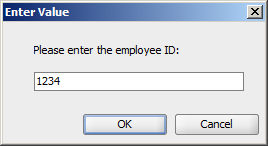
BEGIN

-- *Your statements*

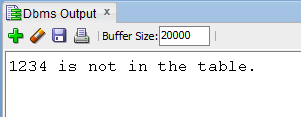
END;

Please test your program. The output of your program must match the following:

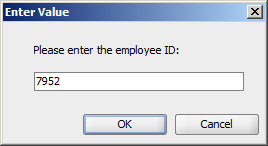
Case 1)



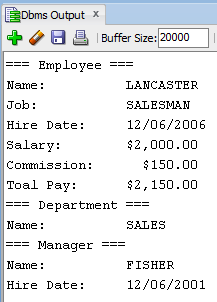
Output:



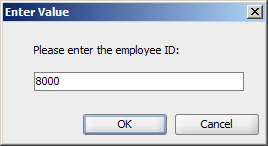
Case 2)



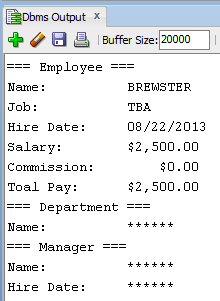
Output:



Case 3)



Output:



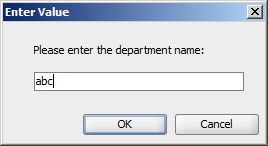
**2) (CSC 352 - 35 points | CSC 452 – 25 points)**

Based on the DEPARTMENT and EMPLOYEE tables created in Assignment #1, write a **PL/SQL anonymous block** that accepts a **department name** from the user input and displays a) the department ID of that department, b) the address of that department, c) the number of employees working in that department, d) the minimum salary for that department, e) the maximum salary for that department, f) the average salary for that department, g) the number of employees in that department whose salary is higher than the average salary for that department, and h) the number of employees in that department whose salary is higher than the average salary of all employees in the **company**.

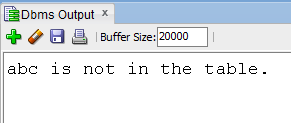
* Submitting more than one PL/SQL program will receive 0 points.
* If the department name from the user input is not in the DEPARTMENT table (DEPARTMENT\_NAME), you display a message telling the user that the department name is not in the table.
* Department’s name is not case sensitive (e.g., SALES = Sales). You will lose 5 points if you do not use the UPPER (or LOWER) function in your program.

Please test your program. The output of your program must match the following:

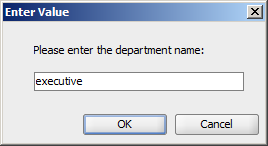
Case 1) (5 points)



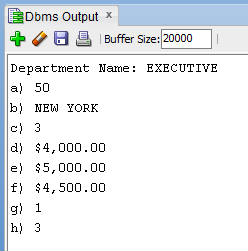
Output:



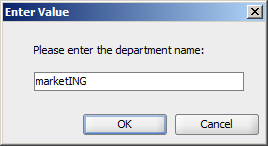
Case 2) (25 points)



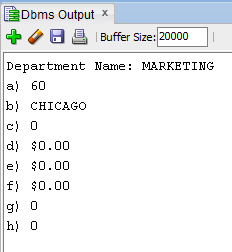
Output:



Case 3) (5 points)



Output:



**3) (CSC 452 only - 15 points)**

Let Fn be a sequence of numbers defined by F0 = 0, F1 = 1, and Fn = Fn-1 + Fn-2, n >= 2.

(F0=0, F1=1, F2=1, F3=2, F4=3, F5=5, F6=8, F7=13, …)

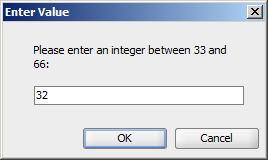
Write a **PL/SQL anonymous block** that accepts an integer n (**33 ≤ n ≤ 66**) from the user input and displays Fn-2, Fn, andFn+2.

* Submitting more than one PL/SQL program will receive 0 points.
* If the user enters a number outside the range of 33 through 66, you display an error message.
* If the user enters 33, you program must display F31, F33, and F35.
* If the user enters 66, you program must display F64, F66, and F68.

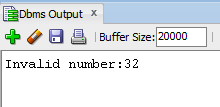
Please test your program.

The output of your program must match the following:

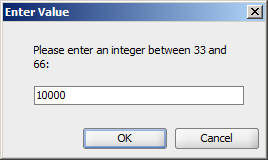
Case 1a)



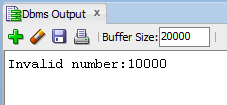
Output:



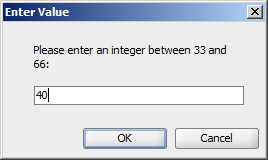
Case 1b)



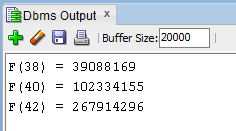
Output:



Case 2)



Output:



Case 3)

……

**Please submit a text file containing all the source codes to D2L before or on due date.**

**Optional Question**

**Just for fun (no credit, no extra credit, no need to submit, just for if you are a curious person and like database programming).**

In a bug tracking database, there is a table called BUGS. The table has several columns: BUG\_ID, REPORTED\_DATE, DESCRIPTION, PRIORITY, ASSIGNED\_TO, CLOSED\_DATE, and NOTE.

Create and populate the BUGS table.

CREATE TABLE bugs

(

BUG\_ID NUMBER PRIMARY KEY,

REPORTED\_DATE DATE NOT NULL,

DESCRIPTION VARCHAR2(20),

PRIORITY NUMBER(2),

ASSIGNED\_TO VARCHAR2(10),

CLOSED\_DATE DATE,

NOTE VARCHAR2(20)

);

INSERT INTO BUGS VALUES (1230, '25-APR-14', NULL, 3, 'Team 3', '28-APR-14', NULL);

INSERT INTO BUGS VALUES (1231, '29-APR-14', NULL, 1, 'Team 1', '29-APR-14', NULL);

INSERT INTO BUGS VALUES (1232, '03-MAY-14', NULL, 1, 'Team 1', '03-MAY-14', NULL);

INSERT INTO BUGS VALUES (1233, '03-MAY-14', NULL, 1, 'Team 3', '08-MAY-14', NULL);

INSERT INTO BUGS VALUES (1234, '04-MAY-14', NULL, 2, 'Team 5', '15-MAY-14', NULL);

INSERT INTO BUGS VALUES (1235, '04-MAY-14', NULL, 2, 'Team 1', NULL, NULL);

INSERT INTO BUGS VALUES (1236, '05-MAY-14', NULL, 1, 'Team 2', '06-MAY-14', NULL);

INSERT INTO BUGS VALUES (1237, '05-MAY-14', NULL, 3, 'Team 3', '10-MAY-14', NULL);

INSERT INTO BUGS VALUES (1238, '09-MAY-14', NULL, 4, 'Team 5', '16-MAY-14', NULL);

INSERT INTO BUGS VALUES (1239, '09-MAY-14', NULL, 5, 'Team 6', NULL, NULL);

INSERT INTO BUGS VALUES (1240, '12-MAY-14', NULL, 5, 'Team 2', '30-MAY-14', NULL);

INSERT INTO BUGS VALUES (1241, '12-MAY-14', NULL, 1, 'Team 1', '20-MAY-14', NULL);

INSERT INTO BUGS VALUES (1242, '13-MAY-14', NULL, 4, 'Team 4', '25-MAY-14', NULL);

INSERT INTO BUGS VALUES (1243, '14-MAY-14', NULL, 4, 'Team 3', '01-JUN-14', NULL);

INSERT INTO BUGS VALUES (1244, '14-MAY-14', NULL, 2, 'Team 4', '25-MAY-14', NULL);

INSERT INTO BUGS VALUES (1245, '20-MAY-14', NULL, 2, 'Team 4', NULL, NULL);

INSERT INTO BUGS VALUES (1246, '22-MAY-14', NULL, 2, 'Team 4', '25-MAY-14', NULL);

INSERT INTO BUGS VALUES (1247, '25-MAY-14', NULL, 2, 'Team 1', '29-MAY-14', NULL);

INSERT INTO BUGS VALUES (1248, '30-MAY-14', NULL, 1, 'Team 1', '01-JUN-14', NULL);

INSERT INTO BUGS VALUES (1249, '05-JUN-14', NULL, 1, 'Team 2', '07-JUN-14', NULL);

COMMIT;

“Open Bugs” - A bug is considered open on a given day if (1) its “REPORTED\_DATE” is on or before that day, and (2) its “CLOSED\_DATE” is on or after that day (or is unknown (NULL)). For example, we have 5 open bugs on 5/5/2014.

Write a **PL/SQL anonymous block** that generates a report to show the number of open bugs from 5/1/2014 through 5/31/2014. At the end of the report, the maximum number of open bugs on a single day is displayed. Assume that there were no open bugs on 4/30/2014.

The output of your program should match the following:

Date Number of Open Bugs

01-MAY-14 0

02-MAY-14 0

03-MAY-14 2

04-MAY-14 3

05-MAY-14 5

06-MAY-14 5

07-MAY-14 4

08-MAY-14 4

09-MAY-14 5

10-MAY-14 5

11-MAY-14 4

12-MAY-14 6

13-MAY-14 7

14-MAY-14 9

15-MAY-14 9

16-MAY-14 8

17-MAY-14 7

18-MAY-14 7

19-MAY-14 7

20-MAY-14 8

21-MAY-14 7

22-MAY-14 8

23-MAY-14 8

24-MAY-14 8

25-MAY-14 9

26-MAY-14 6

27-MAY-14 6

28-MAY-14 6

29-MAY-14 6

30-MAY-14 6

31-MAY-14 5

---------------------------------------

The maximum number of open bugs on a single day is 9.

There were 9 open bugs on 14-MAY-14.

There were 9 open bugs on 15-MAY-14.

There were 9 open bugs on 25-MAY-14.