Midterm (CSC352, CSC452)
Submission closes at 11:00 pm 4/29/2020

Please submit your midterm exam on D2L as you used to submit your homework assignments. But this time, the file(s) formats of doc, docx, rtf or txt format are allowed.

352 and 452 students share the same questions to avoid confusing.
When grading, the instructor will be lenient to student in 352 as requested by university policy.

Part I. True or False. Please circle either True or false for following questions
(2 points for each question)

1) (True, False) A variable cemp is defined as data type of employees%ROWTYPE. We will say, this cemp is a composite variable. In the execution section, the program may use cemp.salary to refer to the correspondent column salary in table of employees.

2) (True, False) System will automatically create a cursor for your DML statement. The name of the cursor will be "SQL", and its attributes can be accessed, such as SQL%ROWCOUNT.

3) If a select statement with a condition such as where employee_id = 100, and we know employee_id has an unique constraint, then we are safe to use the statement of "select into" clause.

4) In an anonymous block, there is an update statement. In that program, later it raises an exception, and that exception is handled. As a result of the exception, this update will be automatically roll-back by the system.

5) (True, False) The command of "PRAGMA EXCEPTION_INIT (*exception_name*, *error_code*)" must be placed in the declaration section.

6) (True, False) Boolean data type is only available in PL/SQL, not in SQL.

7) (True, False) In the execution section of the outer block, the code can refer to the variables that are declared in its enclosed (inner) block.

8) (True, False) In PL/SQL program, each block may have its own Exception section. An inner block can have its exception handler, and the outer block can have its exception section.

9) (True, False) In an exception section, the program can handle several different exceptions, using more than one "WHEN exception_name THEN".

10) (True, False) In one PL/SQL block, the program defines a variable called "sal", after execution of that block, we can refer that variable "sal" again as long as we are in the same session (same log in window of SQL developer).

Part II. Multiple choices, choose one and only one which is the wrong statement, or the least correct. Please read carefully, 5 points for each question.

1)

(a) In the PL/SQL program, the exception handler cannot handle the error occurs in the declaration section in the same block.

(b) The exception handler in exception section can handle the exception that occurs in the execution section in the same block.

(c) The exception handler cannot handle exception which occurs in the same exception section.

(d) A program will get an error and fail to pass the compilation because it has no exception section.

(e) If an exception occurs in the inner block, and it is not handled by the exception section in that inner block, then this exception will propagate to the exception section in its outer (enclosing) block (if there is an outer block).

2)

(a) Programmer can use Pragma Exception_init command to associate a name to an internal defined exception that has only error code (without a name).

(b) When user declares a user-defined exception, originally it does not have its own special error code.

(c) Every system defined exception has a code (number).

(d) Every system internally defined error has an error message.

(e) Every system defined exception has a name, thus it can be raised by programmer.

Part III.

1) (5 points) Following given codes will not pass the compilation, please debug. You need to correct the syntax errors and make the program runnable.

```
DECLARE
   CURSOR c  IS
        SELECT  EMPNO, ENAME
        FROM  EMP ;
     v_c       c%ROWTYPE;

BEGIN
   OPEN  c;
```

```
    LOOP
       FETCH  c INTO V_C;

       DBMS_OUTPUT.PUT_LINE (c%ROWCOUNT ||': Emp ID:  || c.empno ||
        ', name: ' || c.ename) ;
       EXIT WHEN c%ROWCOUNT > 4;
    END LOOP;
    CLOSE  c;

    DBMS_OUTPUT.PUT_LINE('--------------');
END;
```

2) (5 points). Following given codes will not pass the compilation. You need debug, correct the syntax errors and make the program successfully print out the result.

```
DECLARE
      y NUMBER := 1;
BEGIN

For y in 2..5 LOOP

    DBMS_OUTPUT.PUT_LINE ('For loop: y = ' || y);
    y := y + 1 ;
END LOOP;
      DBMS_OUTPUT.PUT_LINE ('WHILE LOOP');

END;
```

Part IV. Write short programs as requested.

1. (12 points)
   Change the program from using basic loop to cursor for loop.

```
DECLARE
CURSOR c IS
      SELECT employee_id, last_name, first_name, salary
      FROM   employees
      ORDER  by 4 desc ;

      c_row  c%ROWTYPE;
BEGIN

      DBMS_OUTPUT.PUT_LINE ('ID   Full name             Salary' );
      DBMS_OUTPUT.PUT_LINE ('---- ------------------    --------');

      OPEN c;
      LOOP
         FETCH c INTO c_row;
         EXIT when c%NOTFOUND or c%ROWCOUNT = 6;
               DBMS_OUTPUT.PUT_LINE ( RPAD (c_row.employee_id, 5) ||
           RPAD ( c_row.first_name ||' '|| c_row.last_name, 20) || ' ' ||
             TO_CHAR (c_row.salary, '$99,999') );
```

```
      END LOOP;
      CLOSE c;
END;
```

2. (12 points) This question requires to use cursor for update, based on table employees. Write an anonymous PL/SQL program, it will change the commission percentage (commission_pct) for those:
if their commission_pct is or greater than .28 and salary is greater than 10,000, then decrease their commission_pct 10% (that means the number X will be X * 0.9).

Your program should declare a cursor with "FOR UPDATE" request, thus it can request the system to lock those records retrieved and change these records later.
After update, print out the info about those affected employees, display their ID, last name, department ID, salary, both commission_pct before and after the change.
Please add an rollback command before the ending the program, to keep table actually unchanged (good for later course assignments).

3. (12 points) Assume that the company has decided a one-time bonus for the employees in the company (using employees table), the amount is decided as below.

For employees
those their salary > = 10000 then bonus := 300
those their salary < 10000 and salary >= 3000 then bonus := 400
those their salary >= 2600 and salary < 3000 then bonus := 500
those their salary < 2600 then bonus := 600

Write a program, declare a cursor that has a parameter. The cursor will retrieve employee ID, last name, salary for those who work for that special department. The department ID will be defined as cursor parameter (IN mode). In the PL/SQL block, you will open the cursor with a value of 30 as department id. Your program will print out the employee's ID, last name, salary and bonus for each employee that work in that department.

4. (12 points)
The program below works fine for current input, but it has no exception handler now. Assume that the value of department_ID will be input by the user, thus this program needs to prepare some strange input value such as `'ABC'` instead of `IT`. Please add an exception section to this program. It is required that using "`WHEN OTHERS`" to capture the error, also using `the functions SQLCODE, SQLERRM` to get the possible error information, then printing out these info.

```
DECLARE
   dept_id        departments.department_id%TYPE;
   mgr_id         departments.manager_id%TYPE;
BEGIN
   SELECT   department_ID, Manager_ID
```

```
      INTO   dept_id, mgr_id
      FROM   departments
      Where  Department_name = 'IT';
                     -- change the value of IT to 'ABC'

    DBMS_OUTPUT.PUT_LINE ('The department ID is: ' || dept_id ||
                  '; Manager ID is: ' || mgr_id);
END;
```

5. (12 points) Below is a draft PL/SQL block. Currently the variable lname has initial value `'Khoo'`. The program below runs fine now.
   (a) Change the initial value of lname from Khoo to 'Grant', run the program again;
   (b) This time, change the initial value of lname to 'XYZ', run the program again.
   In the above (a) and (b) cases, you will get error messages. Please add the exception section to the program below. There should have two exception handlers, each will capture the error in case (a) and (b) individually. The program should print out a proper message for each case. Do not use the "OTHERS" in this question.

```
Declare
    lname     employees.last_name%TYPE := 'Khoo';
                   -- you can change this initial value as if that is the user input
    fname     employees.first_name%TYPE  ;
    sal       employees.salary%TYPE;

BEGIN
   SELECT first_name, salary INTO fname, sal
   FROM   employees
   WHERE  last_name = lname;        -- you may use &dummy_name if you like

   DBMS_OUTPUT.PUT_LINE ('that employee has first name ' ||
            fname || '; Salary is: '|| TO_CHAR(sal, '$99,999'));
END;
```