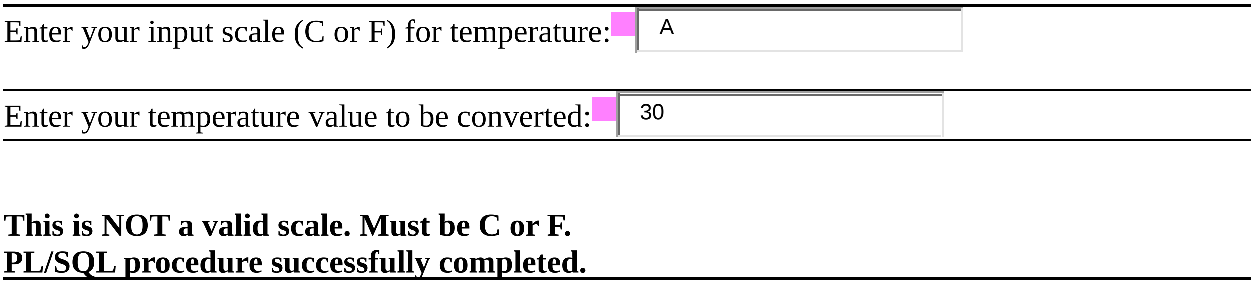
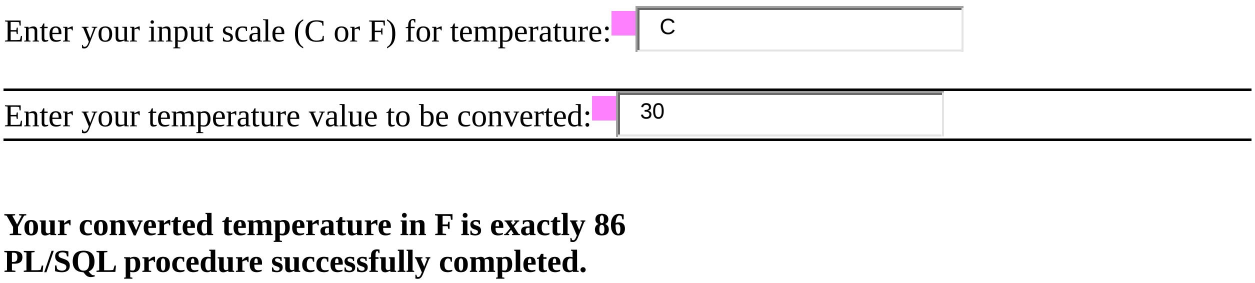
Lab2 due on Friday, Oct 05th by 3pm

Just hand-in the PRINTOUT of your answers. No e-mails.

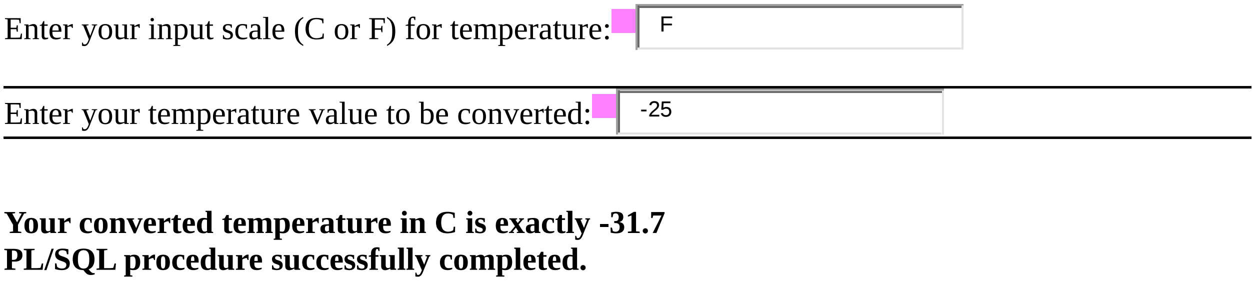
1) Write the PL/SQL code that may be used as a temperature system converter -- you will enter firstly your current (input) scale as C or F and then the value to be converted. The output value should be rounded to only one decimal. Use IF .. THEN logic. Here are the outputs:



```
Enter your input scale (C or F) for temperature: A

Enter your temperature value to be converted: 30

This is NOT a valid scale. Must be C or F.
PL/SQL procedure successfully completed.
```

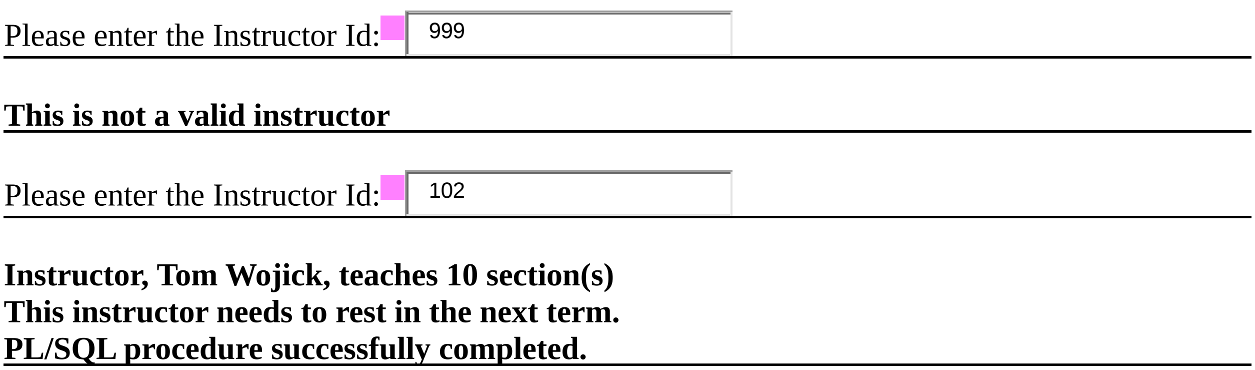


```
Enter your input scale (C or F) for temperature: C

Enter your temperature value to be converted: 30

Your converted temperature in F is exactly 86
PL/SQL procedure successfully completed.
```



```
Enter your input scale (C or F) for temperature: F

Enter your temperature value to be converted: -25

Your converted temperature in C is exactly -31.7
PL/SQL procedure successfully completed.
```

2) Write a PL/SQL block that will ask for Instructor's Id and then will figure out how many sections does teach that person. It will display then an appropriate message based on that number. Use CASE .. WHEN logic. Here are the possible outputs:



```
Please enter the Instructor Id: 999

This is not a valid instructor

Please enter the Instructor Id: 102

Instructor, Tom Wojick, teaches 10 section(s)
This instructor needs to rest in the next term.
PL/SQL procedure successfully completed.
```

Please enter the Instructor Id: 101

```
Instructor, Fernand Hanks, teaches 9 section(s)
This instructor teaches enough sections.
PL/SQL procedure successfully completed.
```

Please enter the Instructor Id: 109

```
Instructor, Rick Chow, teaches 0 section(s)
This instructor may teach more sections.
PL/SQL procedure successfully completed.
```

3) Write a PL/SQL block that will ask for a Positive Integer input and then will calculate the sum of all Even (or Odd) integers between 1 and that value, depending whether the input value is Even or Odd. Use WHILE LOOP control logic. Here are the possible outputs:

Please enter a Positive Integer: 12

```
The sum of Even integers between 1 and 12 is 42
PL/SQL procedure successfully completed.
```

Please enter a Positive Integer: 901

```
The sum of Odd integers between 1 and 901 is 203401
PL/SQL procedure successfully completed.
```

4) Write a PL/SQL block that will ask for department’s Location Id and then will figure out how many departments exist there and how many employees work on that location. Based on these answers you will construct a Nested For Loop that will display these answers in a numeric format like shown below. The first location 1800 has only one department with 2 employees, while the other location 1400 has 3 departments with 7 employees altogether (after the Update).
Obviously, without an explicit cursor you can NOT show how many employees works in each department. So, this is just an exercise in creating nested loops.
But, firstly you need to relocate departments 40 and 70 to location 1400.
Do NOT commit, you will rollback it when you finish your block.

```
UPDATE  Departments

SET     location_id = 1400

WHERE  department_id IN (40,70)
```

```
Enter valid Location Id: 1800

Outer Loop: Department #1
* Inner Loop: Employee #1
* Inner Loop: Employee #2
PL/SQL procedure successfully completed.

Enter valid Location Id: 1400

Outer Loop: Department #1
* Inner Loop: Employee #1
* Inner Loop: Employee #2
* Inner Loop: Employee #3
* Inner Loop: Employee #4
* Inner Loop: Employee #5
* Inner Loop: Employee #6
* Inner Loop: Employee #7
Outer Loop: Department #2
* Inner Loop: Employee #1
* Inner Loop: Employee #2
* Inner Loop: Employee #3
* Inner Loop: Employee #4
* Inner Loop: Employee #5
* Inner Loop: Employee #6
* Inner Loop: Employee #7
Outer Loop: Department #3
* Inner Loop: Employee #1
* Inner Loop: Employee #2
* Inner Loop: Employee #3
* Inner Loop: Employee #4
* Inner Loop: Employee #5
* Inner Loop: Employee #6
* Inner Loop: Employee #7
PL/SQL procedure successfully completed.

ROLLBACK

Rollback complete.
```