DBS Labsheet-8 (CMS)

(Prof.R Gururaj)

PL-SQL Stored Procedures (continued)

Commenting in PLSQL:

PL/SQL has two comment styles: single-line and multi-line comments.

A **single-line** comment starts with a double hyphen (--) that can appear anywhere on a line and extend to the end of the line.

A **multi-line** comment starts with a slash-asterisk (/*) and ends with an asterisk slash (*/), and can span multiple lines:

Concept of IN / OUT / IN OUT Parameters in procedures

IN mode:

- Default mode
- Passes a value to the subprogram.
- Formal parameter acts like a constant: When the subprogram begins, its value is that of either its actual parameter or default value, and the subprogram cannot change this value.
- Actual parameter can be a constant, initialized variable, literal, or expression.

`OUT` mode:

- Must be specified.
- Returns a value to the invoker.
- Formal parameter is initialized to the default value of its type. The default value of the type is `NULL` except for a record type with a non-`NULL` default value.
- When the subprogram begins, the formal parameter has its initial value regardless of the value of its actual parameter. Oracle recommends that the subprogram assign a value to the formal parameter.

`IN OUT` mode:

- Must be specified.
- Passes an initial value to the subprogram and returns an updated value to the invoker.

// We write programs to demonstrating IN and OUT parameters

IF THEN ELSE ladder in PLSQL

```
IF condition1 THEN

   {...statements to execute when condition1 is TRUE...}


ELSIF condition2 THEN

   {...statements to execute when condition1 is FALSE and condition2 is
TRUE...}


ELSE

   {...statements to execute when both condition1 and condition2 are
FALSE...}


END IF;
```

Example

// Procedure to print the grade if marks are given as argument
//Less than 50 ordinary grade; 50-69 First grade; 70 and above Distinction grade

```
SQL> create or replace procedure proc7(marks in number) as
 begin
   if n < 50 then
      dbms_output.put_line(' Ordinary Grade:  ');
   elsif n<70 then
       dbms_output.put_line(' First Grade:  ');
    else
       dbms_output.put_line(' Distinction Grade:  ');
    end if;
  end;
/
Procedure created.
```

Looping in PLSQL

We understand how looping structures can be written in PL-SQL.

Pl-SQL Functions

How Functions are different from Stored Procedures

- **Functions**: these subprograms return a single value, mainly used to compute and return a value.
- **Procedures**: these subprograms do not return a value directly, mainly used to perform an action. Or executing a set of data manipulation operations in one go at DB server.

A PL/SQL function is same as a procedure except that it returns a value. Therefore, all the discussions of the previous chapter are true for functions too.

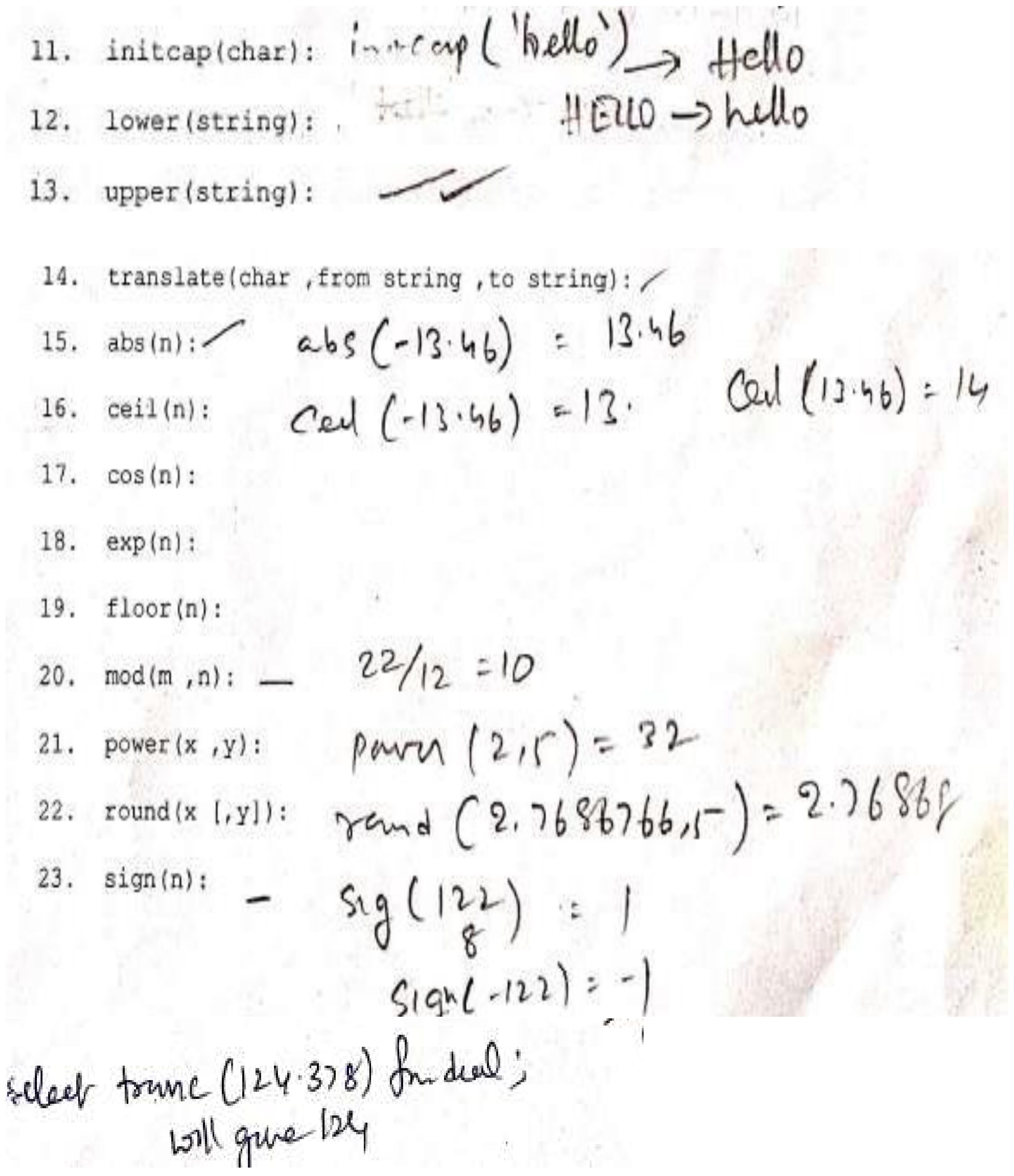
Example

```
SQL> create function Func1(bookid in number) return number is
  prc number;
  begin
  select price into prc from book where bid=bookid;
  return prc;
  end;
  /
Function created.

// write anonymous code calling the function.
SQL> declare
  n number;
  begin
  n:=Func1(107);
  dbms_output.put_line('  '||' price is   : '||n);
  end;
  /
```

We write a function to take two numbers and return the sum

Some more PL-SQL Built in Functions



11. initcap(char): initcap('hello') → Hello

12. lower(string): HELLO → hello

13. upper(string): ✓

14. translate(char ,from string ,to string): ✓

15. abs(n): ✓ abs(-13.46) = 13.46

16. ceil(n): Ceil(-13.46) = -13. Ceil(13.46) = 14

17. cos(n):

18. exp(n):

19. floor(n):

20. mod(m ,n): — 22/12 = 10

21. power(x ,y): power(2,5) = 32

22. round(x [,y]): round(2.76867766,5) = 2.76868

23. sign(n): — sig(1228) : 1

sign(-122) : -1

select trunc(124.378) from dual;
will give 124