DMDD Assignment 3: Tic – Tac – Toe Game

By: Aditya Agarwal
Github Link - https://github.com/adiag321/Tic-Tac-Toe-Game-Using-PL-SQL

Rules:

1. The Row number and column number are indexed with “1”, “2” and “3” and is converted into “A”, “B” and “C”.
2. In this game, all the values are manually inserted, and 4 test cases are implemented
3. “X” and “O” values are inserted in Alternate Fashion.
4. If user inserts values other than “X” and “O”, then an error is shown in the command terminal.
5. If the user inserts row value and column value other than “1”, “2” and “3”, then an error is shown accordingly to the user in the command line.
6. The user needs to restart the game by “Resetting” it and calling the “Resetting_game” procedure.

Code:

```
--drop table tic_tac_toe_game;
set serveroutput on;

CREATE TABLE tic_tac_toe_game(
Y NUMBER,
A CHAR,
B CHAR,
C CHAR
);

select * from tic_tac_toe_game;


CREATE OR REPLACE FUNCTION num_to_col (num IN NUMBER)
RETURN CHAR IS
BEGIN
  IF num = 1 THEN
    RETURN 'A';
  ELSIF num = 2 THEN
    RETURN 'B';
  ELSIF num = 3 THEN
    RETURN 'C';
  ELSE
    RETURN '_';
  END IF;
END;
```

```
CREATE OR REPLACE PROCEDURE show_matrix IS
BEGIN
  dbms_output.put_line(' ');
  FOR i in (SELECT * FROM tic_tac_toe_game ORDER BY Y) LOOP
    dbms_output.put_line('     ' || i.A || ' ' || i.B || ' ' || i.C);
  END LOOP;
  dbms_output.put_line(' ');
END;


CREATE OR REPLACE PROCEDURE reseting_game IS
i NUMBER;
BEGIN
  DELETE FROM tic_tac_toe_game;
  FOR i in 1..3 LOOP
    INSERT INTO tic_tac_toe_game VALUES (i,'_','_','_');
  END LOOP;
  show_matrix();
  dbms_output.put_line('New Game : EXECUTE game(''X'', x, y);');
END;


CREATE OR REPLACE PROCEDURE show_winner (Character_value IN VARCHAR2) IS
BEGIN
  show_matrix();
  dbms_output.put_line('Player ' || Character_value || ' Won the Game.');
  dbms_output.put_line('To start a new game, RESET the game');
END;



CREATE OR REPLACE PROCEDURE playing_game(Character_value IN VARCHAR2, row_value IN
NUMBER, col_value IN NUMBER) IS
val tic_tac_toe_game.a%type;
cols CHAR;
Character_value2 CHAR;
A1 tic_tac_toe_game.a%type;
A2 tic_tac_toe_game.a%type;
A3 tic_tac_toe_game.a%type;
B1 tic_tac_toe_game.a%type;
B2 tic_tac_toe_game.a%type;
B3 tic_tac_toe_game.a%type;
C1 tic_tac_toe_game.a%type;
C2 tic_tac_toe_game.a%type;
```

```
C3 tic_tac_toe_game.a%type;

-- Exception
condition_tie exception;
wrong_row_no exception;
wrong_column_no exception;
wrong_symbol exception;
FLAG BOOLEAN:=TRUE;

BEGIN

-- Handling irregular column value, row value and character
  IF col_value < 1 OR col_value > 3 then
    raise wrong_column_no;
  END IF;
  IF row_value < 1 OR row_value > 3 then
    raise wrong_row_no;
  END IF;
  IF Character_value <> 'O' AND Character_value <> 'X' then
    raise wrong_symbol;
  END IF;

  SELECT num_to_col (col_value) INTO cols FROM DUAL;

  EXECUTE IMMEDIATE ('SELECT ' || cols || ' FROM tic_tac_toe_game WHERE y=' || row_value)
INTO val;
  IF val='_' THEN
    EXECUTE IMMEDIATE ('UPDATE tic_tac_toe_game SET ' || cols || '=''' || Character_value ||
''' WHERE y=' || row_value);
    IF Character_value = 'X' THEN
      Character_value2 := 'O';
    ELSE
      Character_value2 := 'X';

    END IF;
    show_matrix();

    EXECUTE IMMEDIATE ('SELECT A FROM tic_tac_toe_game WHERE y=1') INTO A1;
    EXECUTE IMMEDIATE ('SELECT B FROM tic_tac_toe_game WHERE y=1') INTO B1;
    EXECUTE IMMEDIATE ('SELECT C FROM tic_tac_toe_game WHERE y=1') INTO C1;
    EXECUTE IMMEDIATE ('SELECT A FROM tic_tac_toe_game WHERE y=2') INTO A2;
    EXECUTE IMMEDIATE ('SELECT B FROM tic_tac_toe_game WHERE y=2') INTO B2;
    EXECUTE IMMEDIATE ('SELECT C FROM tic_tac_toe_game WHERE y=2') INTO C2;
    EXECUTE IMMEDIATE ('SELECT A FROM tic_tac_toe_game WHERE y=3') INTO A3;
```

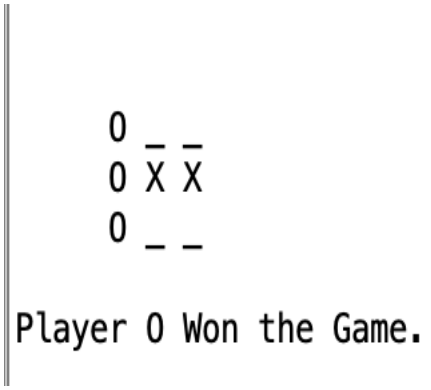
```
   EXECUTE IMMEDIATE ('SELECT B FROM tic_tac_toe_game WHERE y=3') INTO B3;
   EXECUTE IMMEDIATE ('SELECT C FROM tic_tac_toe_game WHERE y=3') INTO C3;

   IF (A1=C1) AND (A1=B1) AND A1 <> '_' THEN
      show_winner(A1);
      FLAG:=FALSE;
   END IF;
   IF (A2=C2) AND (A2=B2) AND A2 <> '_' THEN
      show_winner(A2);
      FLAG:=FALSE;
   END IF;
   IF (A3=C3) AND (A3=B3) AND A3 <> '_' THEN
      show_winner(A3);
      FLAG:=FALSE;
   END IF;
   IF (A1=A3) AND (A1=A2) AND A1 <> '_' THEN
      show_winner(A1);
      FLAG:=FALSE;
   END IF;
   IF (B1=B3) AND (B1=B2) AND B1 <> '_' THEN
      show_winner(B1);
      FLAG:=FALSE;
   END IF;
   IF (C1=C3) AND (C1=C2) AND C1 <> '_' THEN
      show_winner(C1);
      FLAG:=FALSE;
   END IF;
   IF (A1=C3) AND (A1=B2) AND A1 <> '_' THEN
      show_winner(A1);
      FLAG:=FALSE;
   END IF;
   IF (C1=A3) AND (C1=B2) AND C1 <> '_' THEN
      show_winner(C1);
      FLAG:=FALSE;
   END IF;

   IF B1 <> '_' AND B2 <> '_' AND B3 <> '_' AND C1 <> '_' AND C2 <> '_' AND C3 <> '_' AND A1 <>
'_' AND A2 <> '_' AND A3 <> '_'  THEN
         raise condition_tie;
   END IF;
   IF FLAG THEN
      dbms_output.put_line('Next turn ' || Character_value2 || ' to play : EXECUTE game('''' ||
Character_value2 || ''', x, y);');
   END IF;
```

```
  ELSE
    dbms_output.put_line('Other player has played on this slot. Try on different slot');
  END IF;

-- Exeption Handling
EXCEPTION
    WHEN wrong_symbol THEN
        dbms_output.put_line('Given Symbol is not "X" or "O"');
        --reseting_game();

    WHEN condition_tie THEN
        dbms_output.put_line('It is a Tie');
        --reseting_game();

    WHEN wrong_column_no THEN
        dbms_output.put_line('Given Column Number should be in the range of 1 to 3');
        reseting_game();

    WHEN wrong_row_no THEN
        dbms_output.put_line('Given Row Number should be in the range of 1 to 3');
        --reseting_game();
END;
```

TEST CASES:

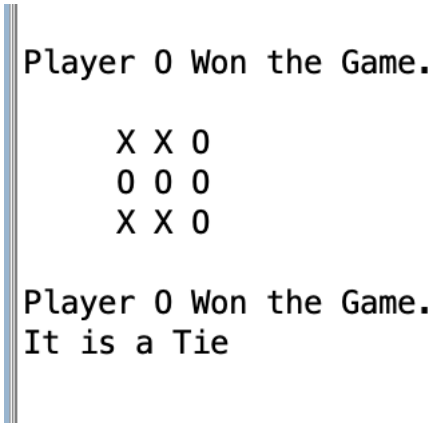
1. Test Case 1 :

```
EXECUTE reseting_game;
EXECUTE playing_game('O', 3, 1);
EXECUTE playing_game('X', 2, 2);
EXECUTE playing_game('O', 1, 1);
EXECUTE playing_game('X', 2, 3);
EXECUTE playing_game('O', 2, 1);
```



2. Test Case 2: Tie Condition

```
EXECUTE reseting_game;
EXECUTE playing_game('O', 1, 3);
EXECUTE playing_game('X', 1, 1);
EXECUTE playing_game('O', 2, 2);
EXECUTE playing_game('X', 3, 1);
EXECUTE playing_game('O', 2, 1);
EXECUTE playing_game('X', 3, 2);
EXECUTE playing_game('O', 2, 3);
EXECUTE playing_game('X', 1, 2);
EXECUTE playing_game('O', 3, 3);
```



3. Test Case 3: UNKNOWN ROW NUMBER AND COLUMN NUMBER

```
EXECUTE reseting_game;
EXECUTE playing_game('X', 1, 3);
EXECUTE playing_game('O', 2, 4);
EXECUTE playing_game('X', 3, 1);
```

EXECUTE playing_game('O', 0, 1);

```
Given Column Number should be in the range of 1 to 3

     _ _ _
     _ _ _
     _ _ _
New Game : EXECUTE game('X', x, y);

PL/SQL procedure successfully completed.


     _ _ _
     _ _ _
     X _ _
Next turn O to play : EXECUTE game('O', x, y);

PL/SQL procedure successfully completed.
Given Row Number should be in the range of 1 to 3
```

4. Test Case 4: WHEN PLAYER INPUTS CHARACTER OTHER THAN "X" AND "O"

EXECUTE reseting_game;
EXECUTE playing_game('X', 1, 3);
EXECUTE playing_game('A', 2, 1);
EXECUTE playing_game('X', 1, 2);
EXECUTE playing_game('O', 2, 3);
EXECUTE playing_game('P', 1, 1);

```
PL/SQL procedure successfully completed.


     _ X X
     _ _ _
     _ _ _
Next turn O to play : EXECUTE game('O', x, y);

PL/SQL procedure successfully completed.


     _ X X
     _ _ O
     _ _ _
Next turn X to play : EXECUTE game('X', x, y);

PL/SQL procedure successfully completed.

Given Symbol is not "X" or "O"
```