**DMDD Assignment 3**

**Yash Nema**

**Assumptions and Facts:**

1. **Game board contains y rows “1”, “2”, “3”, and x columns “A”, “B”, “C” respectively. The x column is converted from numeric from “1”, “2”, “3” to “A”, “B”, “C” respectively**
2. **If user chooses any x or y value other than mentioned above the game resets.**
3. **Game is always started by “X” player**
4. **“X” and “O” Alternate chance, it is assumed order is followed.**
5. **If the user enters input in a cell which already contains data, the user can choose another cell to enter input**
6. **Player is always aware of their turn and current board that has been filled**
7. **Row and column inputs are always integers**
8. **When a game gets over another game starts**

**Code:**

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

--To see the server side print messages in local system

set serveroutput on;

--Creating tic tac toe board

CREATE TABLE TIC\_TAC\_TOE(

Y NUMBER,

A CHAR,

B CHAR,

C CHAR

);

--Creating numeric column number to column name converter

CREATE OR REPLACE FUNCTION num2col(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;

/

--Printing game status-current board

CREATE OR REPLACE PROCEDURE print\_game IS

BEGIN

dbms\_output.put\_line(' ');

FOR i in (SELECT \* FROM TIC\_TAC\_TOE ORDER BY Y) LOOP

dbms\_output.put\_line(' ' || i.A || ' ' || i.B || ' ' || i.C);

END LOOP;

dbms\_output.put\_line(' ');

END;

/

--Reset the game

CREATE OR REPLACE PROCEDURE reset\_game IS

i NUMBER;

BEGIN

DELETE FROM TIC\_TAC\_TOE;

FOR i in 1..3 LOOP

INSERT INTO TIC\_TAC\_TOE VALUES (i,'\_','\_','\_');

END LOOP;

print\_game();

dbms\_output.put\_line('New Game : EXECUTE game(''X'', x, y);');

END;

/

--To show winner of the game

CREATE OR REPLACE PROCEDURE winner(symbol IN VARCHAR2) IS

BEGIN

print\_game();

dbms\_output.put\_line('The Player ' || symbol || ' won!');

dbms\_output.put\_line('Starting a new game...');

reset\_game();

END;

/

--Main code to fetch the user input and show if the user won or needs to continue playing

CREATE OR REPLACE PROCEDURE game(symbol IN VARCHAR2, colnum IN NUMBER, rownum IN NUMBER) IS

val TIC\_TAC\_TOE.a%type;

cols CHAR;

symbol2 CHAR;

A1 TIC\_TAC\_TOE.a%type;

B1 TIC\_TAC\_TOE.a%type;

C1 TIC\_TAC\_TOE.a%type;

A2 TIC\_TAC\_TOE.a%type;

B2 TIC\_TAC\_TOE.a%type;

C2 TIC\_TAC\_TOE.a%type;

A3 TIC\_TAC\_TOE.a%type;

B3 TIC\_TAC\_TOE.a%type;

C3 TIC\_TAC\_TOE.a%type;

tie\_condition exception;

FLAG BOOLEAN:=TRUE;

row\_incorrect exception;

column\_incorrect exception;

symbol\_incorrect exception;

BEGIN

IF rownum<1 OR rownum>3 then

raise row\_incorrect;

END IF;

IF colnum<1 OR colnum>3 then

raise column\_incorrect;

END IF;

IF NOT symbol ='X' AND NOT symbol ='O' then

raise symbol\_incorrect;

END IF;

SELECT num2col(colnum) INTO cols FROM DUAL;

EXECUTE IMMEDIATE ('SELECT ' || cols || ' FROM TIC\_TAC\_TOE WHERE y=' || rownum) INTO val;

IF val='\_' THEN

EXECUTE IMMEDIATE ('UPDATE TIC\_TAC\_TOE SET ' || cols || '=''' || symbol || ''' WHERE y=' || rownum);

IF symbol='X' THEN

symbol2:='O';

ELSE

symbol2:='X';

END IF;

print\_game();

EXECUTE IMMEDIATE ('SELECT A FROM TIC\_TAC\_TOE WHERE y=1') INTO A1;

EXECUTE IMMEDIATE ('SELECT B FROM TIC\_TAC\_TOE WHERE y=1') INTO B1;

EXECUTE IMMEDIATE ('SELECT C FROM TIC\_TAC\_TOE WHERE y=1') INTO C1;

EXECUTE IMMEDIATE ('SELECT A FROM TIC\_TAC\_TOE WHERE y=2') INTO A2;

EXECUTE IMMEDIATE ('SELECT B FROM TIC\_TAC\_TOE WHERE y=2') INTO B2;

EXECUTE IMMEDIATE ('SELECT C FROM TIC\_TAC\_TOE WHERE y=2') INTO C2;

EXECUTE IMMEDIATE ('SELECT A FROM TIC\_TAC\_TOE WHERE y=3') INTO A3;

EXECUTE IMMEDIATE ('SELECT B FROM TIC\_TAC\_TOE WHERE y=3') INTO B3;

EXECUTE IMMEDIATE ('SELECT C FROM TIC\_TAC\_TOE WHERE y=3') INTO C3;

IF (A1=B1) AND (A1=C1) AND NOT A1='\_' THEN

winner(A1);

FLAG:=FALSE;

END IF;

IF (A2=B2) AND (A2=C2) AND NOT A2='\_' THEN

winner(A2);

FLAG:=FALSE;

END IF;

IF (A3=B3) AND (A3=C3) AND NOT A3='\_' THEN

winner(A3);

FLAG:=FALSE;

END IF;

IF (A1=A2) AND (A1=A3) AND NOT A1='\_' THEN

winner(A1);

FLAG:=FALSE;

END IF;

IF (B1=B2) AND (B1=B3) AND NOT B1='\_' THEN

winner(B1);

FLAG:=FALSE;

END IF;

IF (C1=C2) AND (C1=C3) AND NOT C1='\_' THEN

winner(C1);

FLAG:=FALSE;

END IF;

IF (A1=B2) AND (A1=C3) AND NOT A1='\_' THEN

winner(A1);

FLAG:=FALSE;

END IF;

IF (C1=B2) AND (C1=A3) AND NOT C1='\_' THEN

winner(C1);

FLAG:=FALSE;

END IF;

IF NOT A1='\_' AND NOT A2='\_' AND NOT A3='\_' AND NOT B1='\_' AND NOT B2='\_' AND NOT B3='\_' AND NOT C1='\_' AND NOT C2='\_' AND NOT C3='\_' THEN

raise tie\_condition;

END IF;

IF FLAG THEN

dbms\_output.put\_line('Next turn ' || symbol2 || ' to play : EXECUTE game(''' || symbol2 || ''', x, y);');

END IF;

ELSE

dbms\_output.put\_line('You cannot play this square, it is already played');

END IF;

EXCEPTION

WHEN row\_incorrect THEN

dbms\_output.put\_line('Incorrect row input y should be 1 or 2 or 3');

reset\_game();

WHEN column\_incorrect THEN

dbms\_output.put\_line('Incorrect column input x should be 1 or 2 or 3');

reset\_game();

WHEN symbol\_incorrect THEN

dbms\_output.put\_line('Incorrect Symbol input should be X or O only');

reset\_game();

WHEN tie\_condition THEN

dbms\_output.put\_line('Tie condition');

reset\_game();

END;

/

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

**Procedure:**

**Execute create table and then functions and procedures**

**Function NUM2COL compiled**

**Procedure PRINT\_GAME compiled**

**Procedure RESET\_GAME compiled**

**Procedure WINNER compiled**

**Procedure GAME compiled**

**-- Test case 1 – X wins-Column victory**

EXECUTE reset\_game;

EXECUTE game('X', 1, 3);

EXECUTE game('O', 2, 1);

EXECUTE game('X', 1, 2);

EXECUTE game('O', 2, 3);

EXECUTE game('X', 1, 1);

Graphical user interface, text, application, email

Description automatically generated Graphical user interface, text, email

Description automatically generated

**--Test case 2 – O wins -diagonal victory and have shown handling when we try to fill space which already contains data**

EXECUTE reset\_game;

EXECUTE game('X', 1, 3);

EXECUTE game('O', 2, 1);

EXECUTE game('X', 1, 2);

EXECUTE game('O', 1, 1);

EXECUTE game('X', 1, 1);

EXECUTE game('X', 3, 1);

EXECUTE game('O', 2, 2);

EXECUTE game('X', 3, 2);

EXECUTE game('O', 3, 3);

Graphical user interface, text, application, email

Description automatically generated Graphical user interface, text, application, email

Description automatically generated Table

Description automatically generated

--Test case 3 – Tie condition

EXECUTE reset\_game;

EXECUTE game('X', 2, 2);

EXECUTE game('O', 2, 1);

EXECUTE game('X', 1, 2);

EXECUTE game('O', 1, 1);

EXECUTE game('X', 3, 1);

EXECUTE game('O', 3, 2);

EXECUTE game('X', 3, 3);

EXECUTE game('O', 1, 3);

EXECUTE game('X', 2, 3);

Graphical user interface, text, application, email

Description automatically generated Graphical user interface, text, application, email

Description automatically generated Graphical user interface, text, email

Description automatically generated

**--Test case 4 -Invalid column number**

EXECUTE game('X', 4, 3);

Text, email

Description automatically generated

**--Test case 5 -Invalid row number**

EXECUTE game('X', 1, 0);

Text

Description automatically generated

**--Test case 6 -Invalid symbol number**

EXECUTE game('x', 1, 1);

Text

Description automatically generated