* TicTacToe entire code with Test Cases

set serveroutput on;

--create table if it does not exist

declare

var\_tab\_exists int;

begin

select count(table\_name) into var\_tab\_exists from user\_tables where table\_name = 'TICTAC';

if var\_tab\_exists = 0 then

execute immediate ('create table ticTac(

indexTerm number,

P varchar(3),

Q varchar(3),

R varchar(3),

rec\_upd\_gamer\_id varchar(3),

rec\_upd\_ts timestamp default systimestamp

)');

end if;

end;

/

-- Show Grid

create or replace procedure show\_grid as

begin

for i in (select \* from ticTac order by indexTerm) loop

dbms\_output.put\_line(' ' || i."P" || ' ' || i."Q" || ' ' || i."R");

end loop;

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

end;

/

-- Start Game

create or replace procedure start\_game

is

begin

delete from ticTac;

insert into ticTac (indexTerm, P, Q, R) values (1,'\_','\_','\_');

insert into ticTac (indexTerm, P, Q, R) values (2,'\_','\_','\_');

insert into ticTac (indexTerm, P, Q, R) values (3,'\_','\_','\_');

commit;

dbms\_output.put\_line('Run nextTurn() stored procedure to play');

end;

/

-- For returning a character for integer input

create or replace function numberToFieldName(numberEntered in number)

return varchar

is

begin

if numberEntered = 1 then return 'P';

elsif numberEntered = 2 then return 'Q';

elsif numberEntered = 3 then return 'R';

else return '\_';

end if;

end;

/

--for winner

create or replace procedure winner(gamer varchar) as

begin

dbms\_output.put\_line('Gamer '|| gamer || ' Wins!');

show\_grid();

dbms\_output.put\_line('Game will now reset');

start\_game();

end;

/

create or replace procedure checkDiagonals(gamer ticTac.P%type) as

diag1\_1 ticTac.P%type;

diag1\_2 ticTac.P%type;

diag1\_3 ticTac.P%type;

diag2\_1 ticTac.P%type;

diag2\_2 ticTac.P%type;

diag2\_3 ticTac.P%type;

count\_diag number;

begin

count\_diag := 0;

--for 1st Diagonal

select P into diag1\_1 from ticTac where indexTerm = 1;

select Q into diag1\_2 from ticTac where indexTerm = 2;

select R into diag1\_3 from ticTac where indexTerm = 3;

if diag1\_1 = gamer and diag1\_2 = gamer and diag1\_3 = gamer then

count\_diag := 1;

end if;

--for 2nd Diagonal

select P into diag2\_1 from ticTac where indexTerm = 3;

select Q into diag2\_2 from ticTac where indexTerm = 2;

select R into diag2\_3 from ticTac where indexTerm = 1;

if diag2\_1 = gamer and diag2\_2 = gamer and diag2\_3 = gamer then

count\_diag := 1;

end if;

if count\_diag = 1 then

winner(gamer);

end if;

end;

/

create or replace procedure checkRows(gamer ticTac.P%type) as

v1 ticTac.P%type;

v2 ticTac.P%type;

v3 ticTac.P%type;

v4 ticTac.P%type;

v5 ticTac.P%type;

v6 ticTac.P%type;

v7 ticTac.P%type;

v8 ticTac.P%type;

v9 ticTac.P%type;

count\_row number;

begin

count\_row := 0;

--for 1st Row

select P into v1 from ticTac where indexTerm = 1;

select Q into v2 from ticTac where indexTerm = 1;

select R into v3 from ticTac where indexTerm = 1;

if v1 = gamer and v2 = gamer and v3 = gamer then

count\_row := 1;

end if;

--for 2nd Row

select P into v4 from ticTac where indexTerm = 2;

select Q into v5 from ticTac where indexTerm = 2;

select R into v6 from ticTac where indexTerm = 2;

if v4 = gamer and v5 = gamer and v6 = gamer then

count\_row := 1;

end if;

--for 3rd Row

select P into v7 from ticTac where indexTerm = 3;

select Q into v8 from ticTac where indexTerm = 3;

select R into v9 from ticTac where indexTerm = 3;

if v7 = gamer and v8 = gamer and v9 = gamer then

count\_row := 1;

end if;

if count\_row = 1 then

winner(gamer);

end if;

end;

/

create or replace procedure checkColumns(gamer ticTac.P%type) as

v1 ticTac.P%type;

v2 ticTac.P%type;

v3 ticTac.P%type;

v4 ticTac.P%type;

v5 ticTac.P%type;

v6 ticTac.P%type;

v7 ticTac.P%type;

v8 ticTac.P%type;

v9 ticTac.P%type;

count\_col number;

begin

count\_col := 0;

--for 1st Row

select P into v1 from ticTac where indexTerm = 1;

select P into v2 from ticTac where indexTerm = 2;

select P into v3 from ticTac where indexTerm = 3;

if v1 = gamer and v2 = gamer and v3 = gamer then

count\_col := 1;

end if;

--for 2nd Row

select Q into v4 from ticTac where indexTerm = 1;

select Q into v5 from ticTac where indexTerm = 2;

select Q into v6 from ticTac where indexTerm = 3;

if v4 = gamer and v5 = gamer and v6 = gamer then

count\_col := 1;

end if;

--for 3rd Row

select R into v7 from ticTac where indexTerm = 1;

select R into v8 from ticTac where indexTerm = 2;

select R into v9 from ticTac where indexTerm = 3;

if v7 = gamer and v8 = gamer and v9 = gamer then

count\_col := 1;

end if;

if count\_col = 1 then

winner(gamer);

end if;

end;

/

--check Results

create or replace procedure checkResults(gamer ticTac.P%type) as

v\_cnt number;

begin

v\_cnt := 0;

select count(\*) into v\_cnt from (select p, q, r from ticTac where p = '\_' or q = '\_' or r = '\_');

if v\_cnt > 0 then

checkDiagonals(gamer);

checkRows(gamer);

checkColumns(gamer);

else

dbms\_output.put\_line('Game ends in a draw!');

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

start\_game();

end if;

end;

/

create or replace procedure nextTurn(in\_gamer ticTac.P%type, in\_getRowID number, in\_getColID number) is

v\_gamer ticTac.P%type;

v\_fieldVal ticTac.P%type;

v\_getColID varchar(3);

e\_userAlreadyPlayed exception;

begin

select rec\_upd\_gamer\_id into v\_gamer from ticTac where rec\_upd\_ts = (select max(rec\_upd\_ts) from ticTac);

if v\_gamer = in\_gamer then

raise e\_userAlreadyPlayed;

else

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

select numberToFieldName(in\_getColID) into v\_getColID from dual; -- get column

execute immediate ('select ' || v\_getColID || ' from ticTac where indexTerm = ' || in\_getRowID) into v\_fieldVal; --get original record

if upper(in\_gamer) = 'X' or upper(in\_gamer) = 'O' then

if v\_fieldVal = '\_' then

execute immediate ('update ticTac set ' || v\_getColID || '=upper('''|| in\_gamer ||'''), rec\_upd\_gamer\_id = upper('''|| in\_gamer ||'''), rec\_upd\_ts = systimestamp WHERE indexTerm = '|| in\_getRowID);

commit;

else

dbms\_output.put\_line('This square has already been filled');

end if;

else

dbms\_output.put\_line('Wrong gamer entered, please enter either X or O only');

end if;

show\_grid();

checkResults(upper(in\_gamer));

end if;

exception

when e\_userAlreadyPlayed then

dbms\_output.put\_line('You have already played!! Please give a chance to the other gamer');

end;

/

--test case for draw

exec start\_game();

exec nextTurn('X', 1, 1);

exec nextTurn('O', 3, 2);

exec nextTurn('X', 2, 2);

exec nextTurn('O', 3, 3);

exec nextTurn('X', 3, 1);

exec nextTurn('O', 2, 1);

exec nextTurn('X', 1, 2);

exec nextTurn('O', 1, 3);

exec nextTurn('X', 2, 3);

--test case for X wins in diagonal

exec start\_game();

exec nextTurn('X', 1, 1);

exec nextTurn('O', 3, 2);

exec nextTurn('X', 2, 2);

exec nextTurn('O', 3, 1);

exec nextTurn('X', 3, 3);

--test case for O wins in row

exec start\_game();

exec nextTurn('X', 1, 1);

exec nextTurn('O', 3, 2);

exec nextTurn('X', 2, 2);

exec nextTurn('O', 3, 1);

exec nextTurn('X', 2, 1);

exec nextTurn('O', 3, 3);

--test case for X wins in column

exec start\_game();

exec nextTurn('X', 1, 1);

exec nextTurn('O', 3, 2);

exec nextTurn('X', 2, 1);

exec nextTurn('O', 3, 3);

exec nextTurn('X', 3, 1);

--test case for wrong character entered

exec start\_game();

exec nextTurn('X', 1, 1);

exec nextTurn('O', 3, 2);

exec nextTurn('X', 2, 1);

exec nextTurn('O', 3, 3);

exec nextTurn('C', 3, 1);

--test case to show handling of lowercase 'x' or 'o' and starting with gamer O

exec start\_game();

exec nextTurn('o', 1, 1);

exec nextTurn('x', 3, 2);

exec nextTurn('o', 2, 1);

exec nextTurn('x', 3, 3);

exec nextTurn('o', 3, 1);

--test case for a gamer playing two turns at once handling

exec start\_game();

exec nextTurn('X', 1, 1);

exec nextTurn('O', 3, 2);

exec nextTurn('X', 2, 1);

exec nextTurn('O', 3, 3);

exec nextTurn('O', 3, 1);

--test case for a square has already been filled

exec start\_game();

exec nextTurn('X', 1, 1);

exec nextTurn('O', 3, 2);

exec nextTurn('X', 2, 1);

exec nextTurn('O', 3, 3);

exec nextTurn('X', 3, 3);