**1.**

**a)**

/\* Employee Type \*/

CREATE OR REPLACE TYPE EMPLOYEETYPE AS OBJECT

(

Eno NUMBER,

Ename VARCHAR2(50),

EAddress VARCHAR2(50)

) NOT INSTANTIABLE NOT FINAL;

/\* Player Type \*/

CREATE OR REPLACE TYPE PLAYERTYPE UNDER EMPLOYEETYPE

(

PlayingPosition VARCHAR2(50)

) NOT INSTANTIABLE NOT FINAL;

/\* Playing Coach Type \*/

CREATE OR REPLACE TYPE PLAYINGCOACHTYPE UNDER PLAYERTYPE

(

YearsExperience NUMBER

) FINAL;

**b)**

CREATE TABLE Players (

player PlayerType

);

INSERT INTO Players VALUES (

PlayerType( 1, 'Dennis', 'East Lansing', 'Point Guard' ) );

INSERT INTO Players VALUES (

PlayingCoachType( 2, 'Charles', 'East Lansing', 'Forward', 3 ) );

**c)**

ALTER TABLE Players MODIFY COLUMN Player NOT SUBSTITUTABLE AT ALL LEVELS;

d)

Tuples of type PlayerType cannot be inserted into tables of PlayingCoachType.

**2. Incomplete objects only listed:**

create or replace type TeamType as OBJECT

(

name VARCHAR2(50),

players PlayerTableRefType

);

/

create or replace type PlayerScoreType as object

(

player ref PlayerType,

score NUMBER

);

/

**3. Listing of entire GameType object and Body definition:**

create or replace Type GameType as Object

(

gid VARCHAR2(50),

Stadium StadiumType,

team1 REF TeamType,

team2 REF TeamType,

team1Score PlayerScoreTableRefType,

team2Score PlayerScoreTableRefType,

MEMBER FUNCTION getMaxScore RETURN NUMBER,

-- returns highest score of players in the game

MEMBER FUNCTION getWinner RETURN VARCHAR2,

-- return team ID of the winning team of the game.

PRAGMA RESTRICT\_REFERENCES(getMaxScore, WNDS),

PRAGMA RESTRICT\_REFERENCES(getWinner, WNDS)

);

/

create or replace type BODY GameType AS

MEMBER FUNCTION getWinner RETURN VARCHAR2 IS

sum1 NUMBER :=0;

sum2 NUMBER :=0;

score NUMBER;

winner VARCHAR2(10) := NULL;

BEGIN

FOR I IN 1 .. SELF.team1Score.COUNT

LOOP

SELECT DEREF(SELF.team1Score(I)).score INTO score FROM DUAL;

sum1 := sum1+score;

END LOOP;

FOR I IN 1 .. SELF.team2Score.COUNT

LOOP

SELECT DEREF(SELF.team2Score(I)).score INTO score FROM DUAL;

sum2 := sum2+score;

END LOOP;

IF sum1 > sum2 THEN

SELECT DEREF(SELF.team1).name INTO winner FROM DUAL;

ELSIF sum2 > sum1 THEN

SELECT DEREF(SELF.team2).name INTO winner FROM DUAL;

END IF;

RETURN winner;

END getWinner;

MEMBER FUNCTION getMaxScore RETURN NUMBER IS

maxScore NUMBER := 0;

tempScore NUMBER := 0;

BEGIN

FOR I IN 1 .. SELF.team1Score.COUNT

LOOP

SELECT DEREF( SELF.team1Score(I) ).score INTO tempScore FROM DUAL;

IF tempScore > maxScore THEN

maxScore := tempScore;

END IF;

END LOOP;

FOR I IN 1 .. SELF.team2Score.COUNT

LOOP

SELECT DEREF( SELF.team2Score(I) ).score INTO tempScore FROM DUAL;

IF tempScore > maxScore THEN

maxScore := tempScore;

END IF;

END LOOP;

RETURN maxScore;

END getMaxScore;

END;

/

**4.**

**a)**

SELECT DISTINCT T.name

FROM TeamTable T, TABLE( T.players ) TP

WHERE TP.COLUMN\_VALUE.paddr = 'East Lansing';

**b)**

SELECT G.gid, PS.COLUMN\_VALUE.player.pname

FROM Games G, TABLE( G.team1Score MULTISET UNION ALL G.team2Score ) PS

WHERE PS.COLUMN\_VALUE.score=G.getMaxScore();

**c)**

SELECT LT.COLUMN\_VALUE.name Winless\_Team

FROM Leagues L, TABLE( L.lteams ) LT

WHERE NOT EXISTS (

SELECT \*

FROM Games G

WHERE G.getWinner()=LT.COLUMN\_VALUE.name )

AND L.lname='A'