```
MODULE Module1
 CONST robtarget Home:=[[-3.60,214.25,278.82],[3.14662E-5,-0.000509556,-0.999971,0.00753834],
   [0,-1,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
 CONST robtarget Starting_Point:=[[0,0,0],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
   +09]];
 !***********************************************************
 !
 ! Module:  Module1
 !
 ! Description:
 !
 !<Insert description here>
 !
 ! Author: selas
 !
 ! Version: 1.0
 !
 !***********************************************************
 !***********************************************************
 !
 ! Procedure main
 !
 !   This is the entry point of your program
 !
 !***********************************************************

 ! ============ Variables For Draw_Grid ===============
    VAR num Length_Grid;
    VAR robtarget newPoint;
    VAR robtarget currentStartingPoint := Starting_Point;  ! The current starting point for drawing
      the grids
    VAR num gap; ! used also in the Draw_x and Draw_Circle procedures to adjust the shapes
      accordingly
 ! ============ Variables For Draw_X ===============
    VAR num size;
    VAR num Xoffset;
 ! ============ Variables For Draw_Circle ===============
    VAR num radius;
    VAR num CircleOffset;
 ! ============ Variables For SelectSquare ============
    VAR num grid{3,3} := [[0,0,0], [0,0,0], [0,0,0]]; ! 3x3 2D array,representing the three rows and
      three columns of the grid.
    VAR num selectedRow;
    VAR num selectedCol;
    VAR robtarget position;

    VAR bool isPlayerX; ! Defining a boolean variable to know if its player X turn or not.
    VAR num answer;

    PROC main()
        WHILE TRUE DO
```

```
        MoveJ Home, v1000, z50, MyPen\WObj:=Table;
        TPReadNum gap, "Enter the size of square on the grid:";
        Draw_Grid Starting_Point,50;
        TPReadFK answer, "Which shape would like to start", "Cross","Circle", stEmpty, stEmpty,
          stEmpty;
        IF answer = 1 THEN
            isPlayerX := TRUE;  ! Player X starts
            TPWrite "Player X will start the game.";
        ELSEIF answer = 2 THEN
            isPlayerX := FALSE;
            TPWrite "Player O will start the game.";
        ELSE
            ! do nothing
        ENDIF
        WHILE TRUE DO
            ! Show the current player
            IF isPlayerX THEN
                TPWrite "Player X's turn.";
            ELSE
                TPWrite "Player O's turn.";
            ENDIF
            SelectSquare;
            ! Checking for a Win or Draw after every move
            IF CheckWin() THEN
                NewGame;
            ENDIF
        ENDWHILE
    ENDWHILE
ENDPROC
PROC Draw_X(robtarget Xcentre)
    size := gap * 0.6;
    Xoffset := gap * 0.2;
    !Calculate the positions for the "X" based on the starting point, size, and offset
    MoveJ Offs(Xcentre, Xoffset, Xoffset, Xoffset), v1000, fine, MyPen\WObj:=Table;
    MoveL Offs(Xcentre, Xoffset, Xoffset, 0), v500, fine, MyPen\WObj:=Table;
    MoveL Offs(Xcentre, Xoffset + size, Xoffset + size, 0), v500, fine, MyPen\WObj:=Table;
    MoveL Offs(Xcentre, Xoffset + size, Xoffset + size, Xoffset), v500, fine, MyPen\WObj:=Table;
    MoveJ Offs(Xcentre, Xoffset, Xoffset + size, Xoffset), v500, fine, MyPen\WObj:=Table;
    MoveL Offs(Xcentre, Xoffset, Xoffset + size, 0), v500, fine, MyPen\WObj:=Table;
    MoveL Offs(Xcentre, Xoffset + size, Xoffset, 0), v500, fine, MyPen\WObj:=Table;
    MoveL Offs(Xcentre, Xoffset + size, Xoffset, Xoffset), v500, fine, MyPen\WObj:=Table;
    MoveJ offs(currentStartingPoint,0,0,50),v1000,fine,MyPen\WObj:=Table; !Going to the starting
      point after every shape drawing.
ENDPROC
PROC Draw_Circle(robtarget Circlecentre)
    !Calculate the positions for the "O" based on the starting point, radius, and offset
    Circlecentre := Offs(position, gap/2, gap/2, 0);
    radius := gap * 0.4;
    CircleOffset := gap * 0.2;
    MoveJ offs(Circlecentre,-radius,0,CircleOffset),v500,fine,MyPen\WObj:=Table;
    MoveL Offs(Circlecentre,-radius,0,0),v100,fine,MyPen\WObj:=Table;
```

```
    MoveC Offs(Circlecentre,0,-radius, 0),Offs(Circlecentre,radius,0,0),v100,fine,MyPen
      \WObj:=Table;
    MoveC Offs(Circlecentre,0,radius, 0),Offs(Circlecentre,-radius,0,0),v100,fine,MyPen
      \WObj:=Table;
    MoveL Offs(Circlecentre,-radius,0,CircleOffset),v100,fine,MyPen\WObj:=Table;
    MoveJ offs(currentStartingPoint,0,0,50),v500,fine,MyPen\WObj:=Table;!Going to the starting
      point after every shape drawing.
ENDPROC
PROC Draw_Grid(robtarget Gridcentre ,num offset)
    !Calculate the positions for the "grid" based on the starting point and the gap between the
      lines
    Length_Grid := 3 * gap;
    MoveJ offs(Gridcentre,0,gap,offset),v500,fine,MyPen\WObj:=Table;
    MoveL Offs(Gridcentre,0,gap,0),v100,fine,MyPen\WObj:=Table;
    MoveL Offs(Gridcentre,Length_Grid,gap,0),v100,fine,MyPen\WObj:=Table;
    MoveL Offs(Gridcentre,Length_Grid,gap,offset),v100,fine,MyPen\WObj:=Table;
    MoveJ offs(Gridcentre,Length_Grid,2*gap,offset),v500,fine,MyPen\WObj:=Table;
    MoveL Offs(Gridcentre,Length_Grid,2*gap,0),v100,fine,MyPen\WObj:=Table;
    MoveL Offs(Gridcentre,0,2*gap,0),v100,fine,MyPen\WObj:=Table;
    MoveL Offs(Gridcentre,0,2*gap,offset),v100,fine,MyPen\WObj:=Table;
    MoveJ offs(Gridcentre,gap,Length_Grid,offset),v500,fine,MyPen\WObj:=Table;
    MoveL offs(Gridcentre,gap,Length_Grid,0),v500,fine,MyPen\WObj:=Table;
    MoveL offs(Gridcentre,gap,0,0),v500,fine,MyPen\WObj:=Table;
    MoveL offs(Gridcentre,gap,0,offset),v500,fine,MyPen\WObj:=Table;
    MoveJ offs(Gridcentre,2*gap,0,offset),v500,fine,MyPen\WObj:=Table;
    MoveL offs(Gridcentre,2*gap,0,0),v500,fine,MyPen\WObj:=Table;
    MoveL offs(Gridcentre,2*gap,Length_Grid,0),v500,fine,MyPen\WObj:=Table;
    MoveL offs(Gridcentre,2*gap,Length_Grid,offset),v500,fine,MyPen\WObj:=Table;
ENDPROC
PROC SelectSquare()
    TPWrite "Select a square in the grid.";
    TPReadNum selectedRow, "Enter row number (1, 2, or 3):";
    TPReadNum selectedCol, "Enter column number (1, 2, or 3):";
    ! Checking that the user inserted a valid input for the rows and columns.
    IF selectedRow >= 1 AND selectedRow <= 3 AND selectedCol >=1 AND selectedCol <= 3 THEN
        IF grid {selectedRow,selectedCol} = 0 THEN !checking that the chosen square is not
          taken.
            TPWrite "You have chosen a square at Row: "+ NumToStr(selectedRow, 0) + ", Column:"
              + NumToStr(selectedCol, 0);
            ! Calculate position based on row and column
            position := Offs(currentStartingPoint, (selectedRow - 1) * gap, (selectedCol - 1) *
              gap, 0);
                ! assigning a value to the selected square, this enables the program to know
                  which shape is drawn, 1 for X and 2 for O
            IF isPlayerX THEN
                grid {selectedRow,selectedCol} := 1;
                Draw_X position;
            ELSE
                grid {selectedRow,selectedCol} := 2;
                Draw_Circle position;
            ENDIF
```

```
            !Switch the player from O to X and X to O.
            isPlayerX := NOT isPlayerX;
        ELSE
            TPWrite "The square is already taken! please select another square";
            SelectSquare;
        ENDIF
    ELSE
        TPWrite "Invalid input. Please enter values between 1 and 3.";
        SelectSquare;
    ENDIF
ENDPROC
FUNC bool CheckWin()
    VAR bool win := FALSE;
    VAR num winner := 0;
    !Checking for winning Rows or winning Columns, at the end "<> 0" means the square is not
      blank.
    FOR i FROM 1 TO 3 DO
        IF grid{i,1} = grid {i,2} AND grid {i,2} = grid{i,3} AND grid{i,1} <> 0 THEN
            win := TRUE;
            winner := grid {i,1};
        ENDIF
        IF grid{1,i} = grid {2,i} AND grid {2,i} = grid{3,i} AND grid{1,i} <> 0 THEN
            win := TRUE;
            winner := grid {1,i};
        ENDIF
    ENDFOR
    !Checking for winning diagonals, at the end "<> 0" means the square is not blank.
    IF grid{1,1} = grid {2,2} AND grid {2,2} = grid{3,3} AND grid{1,1} <> 0 THEN
        win := TRUE;
        winner := grid {1,1};
    ENDIF
    IF grid{1,3} = grid {2,2} AND grid {2,2} = grid{3,1} AND grid{1,3} <> 0 THEN
        win := TRUE;
        winner := grid {1,3};
    ENDIF
    IF win THEN
        IF winner = 1 THEN
            TPWrite "The winner is: Player X";
        ELSE
            TPWrite "The winner is: Player O";
        ENDIF
        RETURN TRUE;
    ENDIF
    !Checking for a Draw between the players.
    FOR i FROM 1 TO 3 DO
        FOR j FROM 1 TO 3 DO
            IF grid {i,j} = 0 THEN
                RETURN FALSE;
            ENDIF
        ENDFOR
    ENDFOR
```

```
    TPWrite "It's a draw!";
    RETURN TRUE;
ENDFUNC
FUNC robtarget CalculatePosition(num row, num col, num gap)
    position := Offs(Starting_Point, row * gap, col * gap, 0);
    RETURN position;
ENDFUNC
FUNC bool ShiftStartingPoint(robtarget currentPoint)
    VAR num A3paperWidth := 420;
    VAR num A3paperHeight := 297;
    Length_Grid := 3 * gap;
    newPoint := currentPoint;

    ! Move to the next row
    newPoint := Offs(newPoint, Length_Grid + gap, 0, 0);

    ! If the new position exceeds the paper height, reset X and move to the next column
    IF newPoint.trans.x + Length_Grid > A3paperHeight THEN
        newPoint.trans.x := Starting_Point.trans.x;
        newPoint.trans.y := currentPoint.trans.y + (Length_Grid + gap);

        ! If the new column exceeds the paper width, return FALSE (paper is full)
        IF newPoint.trans.y + Length_Grid > A3paperWidth THEN
            RETURN FALSE;
        ENDIF
    ENDIF
    currentStartingPoint := newPoint;
    RETURN TRUE;  ! Paper is not yet full
ENDFUNC
PROC NewGame()
    VAR num newGame;
    VAR bool isExit := FALSE;

    TPReadFK newGame, "Do you want to play another game?", "Yes", "No", stEmpty, stEmpty,
      stEmpty;
    IF newGame = 1 THEN
        TPWrite "Starting a new game!";
        IF ShiftStartingPoint(currentStartingPoint) THEN
            grid := [[0,0,0], [0,0,0], [0,0,0]];
            Draw_Grid currentStartingPoint, 50;

        TPReadFK answer, "Which shape would like to start?", "Cross", "Circle", stEmpty,
          stEmpty, stEmpty;
        IF answer = 1 THEN
            isPlayerX := TRUE;
            TPWrite "Player X will start the game.";
        ELSEIF answer = 2 THEN
            isPlayerX := FALSE;
            TPWrite "Player O will start the game.";
        ENDIF
```

```
            SelectSquare;
            ELSE
                TPWrite "The paper is full, no more space for a new game!";
                MoveJ Home, v1000, z50, MyPen\WObj:=Table;
                isExit := TRUE;
                !EXIT;
            ENDIF
        ELSEIF newGame = 2 THEN
            TPWrite "Thank you for playing!";
            MoveJ Home, v1000, z50, MyPen\WObj:=Table;
            Stop;
        ELSE
            ! Do nothing
        ENDIF
    IF isExit THEN
        Stop;  ! Stop the program
    ENDIF
    ENDPROC
ENDMODULE
```