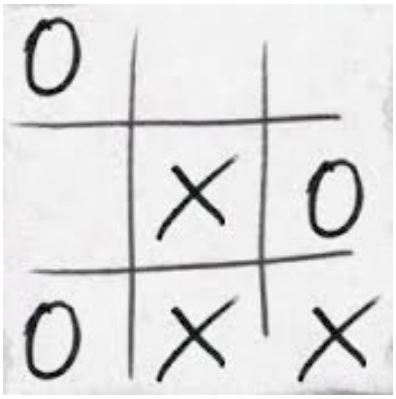
Assignment 1: Fortran (25%)

PLAYING TIC-TAC-TOE

The game of *Tic-Tac-Toe*, known in Britain as *Nought's and Crosses*, is a game played with pencil and paper. Two players, X and O, take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical or diagonal row wins the game.



In this version of the game, a player (X) plays against a computer (O) using a very simple AI engine. The game starts with the user choosing a box to play their X in.

1	2	3
4	5	6
7	8	9

For example, if the user chooses 6, then an "X" is placed in square 6. The computer then plays its turn, placing an "O". After each turn, the board is checked to see if a winner has emerged. If there is a winner, the game ends, otherwise it continues. In 1952, OXO, short for Noughts and Crosses, was developed by British computer scientist Alexander Douglas, and was one of the first computer games.

This topic involves re-engineering a legacy Fortran component, and some building of additional content to augment the legacy re-engineered component, and complete the program.

TASK

Two legacy Fortran subroutines are provided: **`CHKOVR()`** and **`CHKPLAY()`** (written in Fortran 77 or earlier).

`CHKOVR()`
- Determines if there is a winner in a game of Tic-Tac-Toe.

`CHKPLAY()`
- Checks to make sure the human player cannot make a play in a square that is already occupied.

Perform the following:

1. Complete a functioning program, `tictactoe`, which plays a game of Tic-Tac-Toe between a human and a computer.
2. Migrate the code for `CHKOVR()` and `CHKPLAY()` to F95/F2003 standards.
3. Implement any dependencies, eg. the function `same()`.
4. Implement five functions `playTicTacToe()`, `getMove()`, `pickMove()`, `chkplay()`, and `showBoard()`.
5. Test the program thoroughly.

Subprograms dependencies:

`same()`
- A logical function which tests a row, column or diagonal returns a value of true if all three elements are the same ('X' or 'O'); otherwise returns false.

`playTicTacToe()`
- Subroutine which plays the game. Assume one player is human and the other player is a computer. This subroutine would include the humans move, but not the computers move (which is obviously a little more complicated - see function `computerMove()`).

`getMove()`
- Gets the humans move. The output from `getMove()` is a number from 1 to 9 representing the chosen square. Should validate player input.

`pickMove()`
- Performs the computers move. An algorithm for the computer is described below.

`chkplay()`
- Checks to make sure the human player cannot make a play in a square that is already occupied.

`showBoard()`
- Prints out the tic-tac-toe board, each time a player moves.

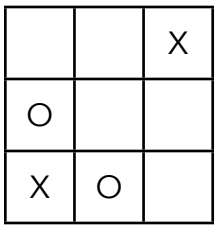
Re-engineering:

Make sure to convert/remove any structures which are relevant/irrelevant. Your code should be clean and easy to understand (unlike the existing code). The program may contain structures which should be modified or removed to make the program more maintainable. Some examples include: GOTO statements, arithmetic IF statements, and label-enabled looping structures. Structures such as arrays, and strings should also be modified to newer constructs.

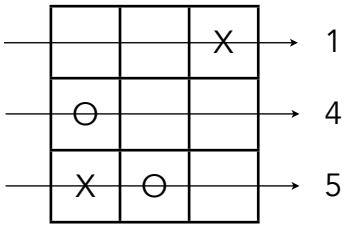
Note that **`CHKOVR()`** and **`CHKPLAY()`** are components of the whole program, and require the use of a data structure **`tictac`**, used to hold the game, and is passed between functions. Do not use a global data structure.

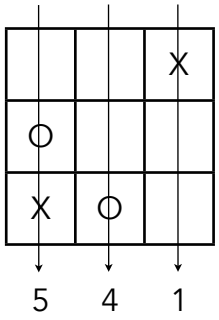
Hint for Computer "AI" Algorithm:

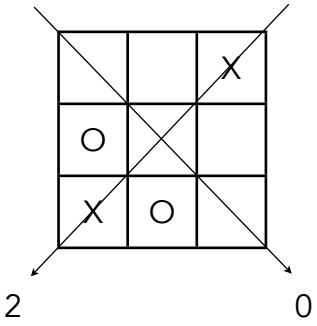
This is a simple "AI" algorithm that performs the computers move (with the assumption that the human player goes first (X), and then the computer (O). It is an algorithm which involves checking the eight paths through the board and deriving a sum for each path based on a value for each of X (1), O (4), or blank (0). For example, the following board:



Would create the following sums:







The algorithm first searches for a sum of 8, which signifies that there are two O's in a path, and then proceeds to add the third O in order to win the game. If this scenario does not work, the algorithm then looks for a sum of 2, signifying a path with two X's and a potential win for the opponent. If one exists, a O is used to block that path. In neither of those situations exist, the algorithm chooses a random spot to place an O.

Sample Interface:

```
PLAY TIC-TAC-TOE. ENTER 1-9 TO PLAY

         1 | 2 | 3
        ---+---+---
         4 | 5 | 6
        ---+---+---
         7 | 8 | 9

 Your move?
4

 After your move...
    |   |
 ---+---+---
  X |   |
 ---+---+---
    |   |

 After my move...
  O |   |
 ---+---+---
  X |   |
 ---+---+---
    |   |

 Your move?
9
...
```

WRITE-UP INFORMATION

REFLECTION REPORT

Discuss your program in a 2 page (or more if you like) **reflection report** (single-spaced), explaining your algorithm and decisions you made in the process of migrating your program. It should provide a synopsis of your experience with Fortran. The write-up should explain your approach to the problem, including any modifications made to the specifications.

You should attempt to answer the following questions:

- What were the greatest problems faced while re-engineering the algorithm in Fortran?
- What particular features make Fortran a good language? (In comparison to C for instance)
- Would it have been easier to write the program in a language such as C?
- Given your knowledge of programming, was Fortran easy to learn?

DELIVERABLES

The submission should consist of the following items:

- The reflection report (PDF).
- The code (well documented and styled appropriately of course). The code should be submitted as a ZIP, TAR, or GZIP file.

SKILLS

- Fortran programming, re-engineering, program comprehension, ability to review specifications, ability to add on additional functionality, ability to write a Fortran program.

SUBROUTINE CHKOVR()

```
      SUBROUTINE CHKOVR(TICTAC, OVER, WINNER)
C
C CHECK IF TIC-TAC-TOE IS OVER AND DETERMINE WINNER (IF ANY)
C
C ARGUMENT DEFINITIONS --
C   INPUT ARGUMENTS
C     TICTAC - REPRESENTS THE CURRENT STATE OF THE BOARD GAME
C   OUTPUT ARGUMENTS
C     OVER - INDICATES WHETHER OR NOT GAME IS OVER
C     WINNER - INDICATES THE WINNER (O OR X) OR A DRAW (D)
C
      CHARACTER * 1 TICTAC(3,3), WINNER
      LOGICAL OVER
C
C SUBROUTINE PARAMETERS
      CHARACTER * 1 BLANK, DRAW
      PARAMETER (BLANK = ' ', DRAW = 'D')
C
C FUNCTIONS USED
      LOGICAL SAME
C
C LOCAL VARIABLES
      LOGICAL DSAME
      INTEGER IR, IC
C ASSUME GAME IS OVER AT START
      OVER = .TRUE.
C
C CHECK FOR A WINNER
C CHECK ROWS FOR A WINNER
      DO 10 IR=1, 3
      IF (SAME(TICTAC(IR,1), TICTAC(IR,2), TICTAC(IR,3))) THEN
          WINNER = TICTAC(IR,1)
          RETURN
      ENDIF
   10 CONTINUE
C NO WINNER BY ROWS, CHECK COLUMNS FOR A WINNER
      DO 20 IC=1, 3
      IF (SAME(TICTAC(1,IC), TICTAC(2,IC), TICTAC(3,IC))) THEN
          WINNER = TICTAC(I,IC)
          RETURN
      ENDIF
   20 CONTINUE
C NO WINNER BY ROWS OR COLUMNS, CHECK DIAGONALS FOR A WINNER
```

```
      DSAME = SAME(TICTAC(1,1), TICTAC(2,2), TICTAC(3,3))
     Z .OR. SAME(TICTAC(1,3), TICTAC(2,2), TICTAC(3,1))
      IF (DSAME) THEN
          WINNER = TICTAC(2,2)
          RETURN
      END IF
C NO WINNER AT ALL. SEE IF GAME IS A DRAW
C CHECK EACH ROW FOR AN EMPTY SPACE
      DO 40 IR = 1, 3
      DO 45 IC = 1, 3
      IF (TICTAC(IR,IC) .EQ. BLANK) THEN
          OVER = .FALSE.
          RETURN
      END IF
   45 CONTINUE
   40 CONTINUE
C
C NO BLANK FOUND, GRAME IS A DRAW
      WINNER = DRAW
C
      RETURN
      END
```

SUBROUTINE CHKPLAY()

```
      LOGICAL FUNCTION CHKPLAY(TICTAC,MOVE)
      CHARACTER * 1 TICTAC(3,3)
      INTEGER MOVE

      GO TO (401,402,403,404,405,406,407,408,409) MOVE
401   IF (TICTAC(1,1) .EQ. " " GOTO 411
      GO TO 410
402   IF (TICTAC(1,2) .EQ. " " GOTO 411
      GO TO 410
403   IF (TICTAC(1,3) .EQ. " " GOTO 411
      GO TO 410
404   IF (TICTAC(2,1) .EQ. " " GOTO 411
      GO TO 410
405   IF (TICTAC(2,2) .EQ. " " GOTO 411
      GO TO 410
406   IF (TICTAC(2,3) .EQ. " " GOTO 411
      GO TO 410
407   IF (TICTAC(3,1) .EQ. " " GOTO 411
      GO TO 410
408   IF (TICTAC(3,2) .EQ. " " GOTO 411
      GO TO 410
409   IF (TICTAC(3,3) .EQ. " " GOTO 411
410   CHKPLAY = .FALSE.
      GOTO 412
411   CHKPLAY = .TRUE.
412   END
```