#include <iostream>

#include <cstdlib>

#include <ctime>

#include <algorithm>

using namespace std;

#define COMPUTER 1

#define HUMAN 2

#define SIDE 3 // Length of the board

// Computer will move with 'O' and human with 'X'

#define COMPUTERMOVE 'O'

#define HUMANMOVE 'X'

// Function to show the current board status

void showBoard(char board[][SIDE]) {

cout << endl << "\t\t\t " << board[0][0] << " | " << board[0][1] << " | " << board[0][2] << endl;

cout << "\t\t\t--------------" << endl;

cout << "\t\t\t " << board[1][0] << " | " << board[1][1] << " | " << board[1][2] << endl;

cout << "\t\t\t--------------" << endl;

cout << "\t\t\t " << board[2][0] << " | " << board[2][1] << " | " << board[2][2] << endl << endl;

}

// Function to show the instructions

void showInstructions() {

cout << "\t\t\t Tic-Tac-Toe" << endl << endl;

cout << "Choose a cell numbered from 1 to 9 as below and play" << endl << endl;

cout << "\t\t\t 1 | 2 | 3" << endl;

cout << "\t\t\t--------------" << endl;

cout << "\t\t\t 4 | 5 | 6" << endl;

cout << "\t\t\t--------------" << endl;

cout << "\t\t\t 7 | 8 | 9" << endl << endl;

cout << "-\t-\t-\t-\t-\t-\t-\t-\t-\t-" << endl << endl;

}

// Function to initialise the game

void initialise(char board[][SIDE], int moves[]) {

srand(time(NULL));

for (int i = 0; i < SIDE; i++) {

for (int j = 0; j < SIDE; j++)

board[i][j] = ' ';

}

for (int i = 0; i < SIDE \* SIDE; i++)

moves[i] = i;

random\_shuffle(moves, moves + SIDE \* SIDE);

}

// Function to declare the winner of the game

void declareWinner(int whoseTurn) {

if (whoseTurn == COMPUTER)

cout << "COMPUTER has won" << endl;

else

cout << "HUMAN has won" << endl;

}

// Function to check if any row is crossed with the same player's move

bool rowCrossed(char board[][SIDE]) {

for (int i = 0; i < SIDE; i++) {

if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0] != ' ')

return true;

}

return false;

}

// Function to check if any column is crossed with the same player's move

bool columnCrossed(char board[][SIDE]) {

for (int i = 0; i < SIDE; i++) {

if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0][i] != ' ')

return true;

}

return false;

}

// Function to check if any diagonal is crossed with the same player's move

bool diagonalCrossed(char board[][SIDE]) {

if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0] != ' ')

return true;

if (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2] != ' ')

return true;

return false;

}

// Function to check if the game is over

bool gameOver(char board[][SIDE]) {

return (rowCrossed(board) || columnCrossed(board) || diagonalCrossed(board));

}

// Function to play Tic-Tac-Toe

void playTicTacToe(int whoseTurn) {

char board[SIDE][SIDE];

int moves[SIDE \* SIDE];

initialise(board, moves);

showInstructions();

int moveIndex = 0, x, y;

while (gameOver(board) == false && moveIndex != SIDE \* SIDE) {

if (whoseTurn == COMPUTER) {

x = moves[moveIndex] / SIDE;

y = moves[moveIndex] % SIDE;

board[x][y] = COMPUTERMOVE;

cout << "COMPUTER has put a " << COMPUTERMOVE << " in cell " << moves[moveIndex] + 1 << endl;

showBoard(board);

moveIndex++;

whoseTurn = HUMAN;

} else if (whoseTurn == HUMAN) {

x = moves[moveIndex] / SIDE;

y = moves[moveIndex] % SIDE;

board[x][y] = HUMANMOVE;

cout << "HUMAN has put a " << HUMANMOVE << " in cell " << moves[moveIndex] + 1 << endl;

showBoard(board);

moveIndex++;

whoseTurn = COMPUTER;

}

}

if (gameOver(board) == false && moveIndex == SIDE \* SIDE)

cout << "It's a draw" << endl;

else {

if (whoseTurn == COMPUTER)

whoseTurn = HUMAN;

else if (whoseTurn == HUMAN)

whoseTurn = COMPUTER;

declareWinner(whoseTurn);

}

}

// Driver program

int main() {

playTicTacToe(COMPUTER);

return 0;

}