#include <iostream>

using namespace std;

char board[10] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

int aiTurn = 1;

int difficulty = 2; // Set difficulty to Hard (2)

bool gameEnded = false;

void drawBoard() {

cout << endl << board[1] << " | " << board[2] << " | " << board[3] << endl

<< "- | - | -" << endl

<< board[4] << " | " << board[5] << " | " << board[6] << endl

<< "- | - | -" << endl

<< board[7] << " | " << board[8] << " | " << board[9] << endl << endl;

}

bool check(int location) {

return ((board[location] != 'X' && board[location] != 'O') ? true : false);

}

int randomNum(int number = 9) {

return (rand() % number + 1);

}

int randomCheck(int rnd, int number = 9) {

if (check(rnd)) {

return rnd;

} else {

return randomCheck(randomNum(number));

}

}

int diagonalProbability(int region1, int region2, int location) {

if (((board[region1] == 'X' && board[region2] == 'X') || (board[region1] == 'O' && board[region2] == 'O')) && (board[location] != 'X' && board[location] != 'O'))

return location;

return 0;

}

int ai() {

// Probabilities:

// Vertical Probabilities:

// If 1,2,3 and 4,5,6 are filled with the same type, place in 7,8,9.

for (int a = 1, b = 4, c = 7; a <= 3; a++, b++, c++) {

if (((board[a] == 'X' && board[b] == 'X') || (board[a] == 'O' && board[b] == 'O')) && (board[c] != 'X' && board[c] != 'O'))

return c;

}

// If 7,8,9 and 4,5,6 are filled with the same type, place in 1,2,3.

for (int a = 7, b = 4, c = 1; a <= 9; a++, b++, c++) {

if (((board[a] == 'X' && board[b] == 'X') || (board[a] == 'O' && board[b] == 'O')) && (board[c] != 'X' && board[c] != 'O'))

return c;

}

// If 1,2,3 and 7,8,9 are filled with the same type, place in 4,5,6.

for (int a = 1, b = 7, c = 4; a <= 3; a++, b++, c++) {

if (((board[a] == 'X' && board[b] == 'X') || (board[a] == 'O' && board[b] == 'O')) && (board[c] != 'X' && board[c] != 'O'))

return c;

}

// Horizontal Probabilities:

// If 1,4,7 and 2,4,8 are filled with the same type, place in 3,6,9.

for (int a = 1; a <= 7; a += 3) {

if (((board[a] == 'X' && board[a + 1] == 'X') || (board[a] == 'O' && board[a + 1] == 'O')) && (board[a + 2] != 'X' && board[a + 2] != 'O'))

return (a + 2);

}

// If 2,5,8 and 3,6,9 are filled with the same type, place in 1,4,7.

for (int a = 2; a <= 8; a += 3) {

if (((board[a] == 'X' && board[a + 1] == 'X') || (board[a] == 'O' && board[a + 1] == 'O')) && (board[a - 1] != 'X' && board[a - 1] != 'O'))

return (a - 1);

}

// If 1,4,7 and 3,6,9 are filled with the same type, place in 2,5,8.

for (int a = 1; a <= 7; a += 3) {

if (((board[a] == 'X' && board[a + 2] == 'X') || (board[a] == 'O' && board[a + 2] == 'O')) && (board[a + 1] != 'X' && board[a + 1] != 'O'))

return (a + 1);

}

// Diagonal Probabilities:

// 1 - 5 -> 9

if (diagonalProbability(1, 5, 9) != 0)

return diagonalProbability(1, 5, 9);

// 3 - 5 -> 7

if (diagonalProbability(3, 5, 7) != 0)

return diagonalProbability(3, 5, 7);

// 5 - 9 -> 1

if (diagonalProbability(5, 9, 1) != 0)

return diagonalProbability(5, 9, 1);

// 7 - 5 -> 3

if (diagonalProbability(7, 5, 3) != 0)

return diagonalProbability(7, 5, 3);

// 1 - 9 -> 5

if (diagonalProbability(1, 9, 5) != 0)

return diagonalProbability(1, 9, 5);

// 3 - 7 -> 5

if (diagonalProbability(3, 7, 5) != 0)

return diagonalProbability(3, 7, 5);

// If hard is selected and there is no probability, play in the corner:

int rnd = randomCheck(randomNum(4));

switch (rnd) {

case 1:

return 1;

case 2:

return 3;

case 3:

return 7;

case 4:

return 9;

default:

cout << "An error occurred: If hard is selected and there is no probability, play in the corner." << endl;

break;

}

// If there is no probability, choose randomly:

return randomCheck(randomNum());

}

void aiMove() {

if (aiTurn == 1) {

// If not hard, select a random place in the first move:

if (difficulty != 2) {

int rnd = randomNum();

if (check(rnd)) {

board[rnd] = 'O';

} else {

aiMove();

}

} else {

// If the center is empty:

if ((board[5] != 'X' && board[5] != 'O')) {

board[5] = 'O';

} else {

// If the center is not empty:

board[ai()] = 'O';

}

}

} else {

board[ai()] = 'O';

}

if (difficulty != 0)

aiTurn++;

}

void play(char player) {

if (player == 'X') {

int location;

cout << "It's X's turn: ";

cin >> location;

if (check(location)) {

board[location] = 'X';

} else {

cout << "Invalid input!" << endl;

play('X');

}

} else {

cout << "It's O's turn..." << endl;

aiMove();

}

}

void whoWon(int location) {

if (board[location] == 'X') {

cout << "Winner: X\nCongratulations, you won!" << endl;

} else {

cout << "Winner: O\nArtificial intelligence won!" << endl;

}

gameEnded = true;

}

bool gameFinished() {

// Horizontal Check

for (int i = 1; i <= 7; i += 3) {

if ((board[i] == 'X' && board[i + 1] == 'X' && board[i + 2] == 'X') || (board[i] == 'O' && board[i + 1] == 'O' && board[i + 2] == 'O')) {

whoWon(i + 1);

return true;

}

}

// Vertical Check

for (int i = 1; i <= 9; i++) {

if ((board[i] == 'X' && board[i + 3] == 'X' && board[i + 6] == 'X') || (board[i] == 'O' && board[i + 3] == 'O' && board[i + 6] == 'O')) {

whoWon(i + 3);

return true;

}

}

// Diagonal Check:

if ((board[1] == 'X' && board[5] == 'X' && board[9] == 'X') || (board[1] == 'O' && board[5] == 'O' && board[9] == 'O')) {

whoWon(5);

return true;

}

if ((board[3] == 'X' && board[5] == 'X' && board[7] == 'X') || (board[3] == 'O' && board[5] == 'O' && board[7] == 'O')) {

whoWon(5);

return true;

}

return false;

}

int main() {

int round = 1;

srand(time(NULL));

system("clear");

cout << endl << "Game Started!" << endl;

do {

drawBoard();

play(round % 2 != 0 ? 'X' : 'O');

round++;

if (gameFinished()) {

break;

} else {};

} while (round <= 9);

drawBoard();

if (!gameEnded)

cout << "Good game!\nIt's a draw!\n";

cout << "Game Over!" << endl;

}