#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define BOARD\_ROWS 6

#define BOARD\_COLS 7

void printBoard(char \*board);

int takeTurn(char \*board, int player, const char\*);

int checkWin(char \*board);

int checkFour(char \*board, int, int, int, int);

int horizontalCheck(char \*board);

int verticalCheck(char \*board);

int diagonalCheck(char \*board);

int main(int argc, char \*argv[]){

const char \*PIECES = "XO";

char board[BOARD\_ROWS \* BOARD\_COLS];

memset(board, ' ', BOARD\_ROWS \* BOARD\_COLS);

int turn, done = 0;

for(turn = 0; turn < BOARD\_ROWS \* BOARD\_COLS && !done; turn++){

printBoard(board);

while(!takeTurn(board, turn % 2, PIECES)){

printBoard(board);

puts("\*\*Column full!\*\*\n");

}

done = checkWin(board);

}

printBoard(board);

if(turn == BOARD\_ROWS \* BOARD\_COLS && !done){

puts("It's a tie!");

} else {

turn--;

printf("Player %d (%c) wins!\n", turn % 2 + 1, PIECES[turn % 2]);

}

return 0;

}

void printBoard(char \*board){

int row, col;

//system("clear");

puts("\n \*\*\*\*Connect Four\*\*\*\*\n");

for(row = 0; row < BOARD\_ROWS; row++){

for(col = 0; col < BOARD\_COLS; col++){

printf("| %c ", board[BOARD\_COLS \* row + col]);

}

puts("|");

puts("-----------------------------");

}

puts(" 1 2 3 4 5 6 7\n");

}

int takeTurn(char \*board, int player, const char \*PIECES){

int row, col = 0;

printf("Player %d (%c):\nEnter number coordinate: ", player + 1, PIECES[player]);

while(1){

if(1 != scanf("%d", &col) || col < 1 || col > 7 ){

while(getchar() != '\n');

puts("Number out of bounds! Try again.");

} else {

break;

}

}

col--;

for(row = BOARD\_ROWS - 1; row >= 0; row--){

if(board[BOARD\_COLS \* row + col] == ' '){

board[BOARD\_COLS \* row + col] = PIECES[player];

return 1;

}

}

return 0;

}

int checkWin(char \*board){

return (horizontalCheck(board) || verticalCheck(board) || diagonalCheck(board));

}

int checkFour(char \*board, int a, int b, int c, int d){

return (board[a] == board[b] && board[b] == board[c] && board[c] == board[d] && board[a] != ' ');

}

int horizontalCheck(char \*board){

int row, col, idx;

const int WIDTH = 1;

for(row = 0; row < BOARD\_ROWS; row++){

for(col = 0; col < BOARD\_COLS - 3; col++){

idx = BOARD\_COLS \* row + col;

if(checkFour(board, idx, idx + WIDTH, idx + WIDTH \* 2, idx + WIDTH \* 3)){

return 1;

}

}

}

return 0;

}

int verticalCheck(char \*board){

int row, col, idx;

const int HEIGHT = 7;

for(row = 0; row < BOARD\_ROWS - 3; row++){

for(col = 0; col < BOARD\_COLS; col++){

idx = BOARD\_COLS \* row + col;

if(checkFour(board, idx, idx + HEIGHT, idx + HEIGHT \* 2, idx + HEIGHT \* 3)){

return 1;

}

}

}

return 0;

}

int diagonalCheck(char \*board){

int row, col, idx, count = 0;

const int DIAG\_RGT = 6, DIAG\_LFT = 8;

for(row = 0; row < BOARD\_ROWS - 3; row++){

for(col = 0; col < BOARD\_COLS; col++){

idx = BOARD\_COLS \* row + col;

if(count <= 3 && checkFour(board, idx, idx + DIAG\_LFT, idx + DIAG\_LFT \* 2, idx + DIAG\_LFT \* 3) || count >= 3 && checkFour(board, idx, idx + DIAG\_RGT, idx + DIAG\_RGT \* 2, idx + DIAG\_RGT \* 3)){

return 1;

}

count++;

}

count = 0;

}

return 0;

}