ORIGINAL CHESS.C (WRONG ONE)

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

// ♔ ♕ ♖ ♗ ♘ ♙

/\* ┌ └ ┐ ┘

├ ┤ ┬ ┴

─ │ ┼

might need these...

╟ ╢ ╧ ╤

\*/

void showBoard (int board[8][8]) {

printf("\n\n 01234567\n");

printf(" ╔════════╗\n");

for (int i = 0; i < 8; i++) {

printf("%d║",i);

for (int j = 0; j < 8; j++) {

if (board[i][j] == 0) {

if ((i + j) % 2 == 0) {

printf(" ");

} else {

printf("▒");

}

} else if (board[i][j] == 1) {

printf("♔");

} else if (board[i][j] == 2) {

printf("♕");

} else if (board[i][j] == 3) {

printf("♖");

} else if (board[i][j] == 4) {

printf("♗");

} else if (board[i][j] == 5) {

printf("♘");

} else if (board[i][j] == 6) {

printf("♙");

}

}

printf("║\n") ;

}

printf(" ╚════════╝\n");

}

void makeMove (char\* input, int board[8][8]) {

int start\_y = input[0] - 48;

int start\_x = input[1] - 48;

int end\_y = input[3] - 48;

int end\_x = input[4] - 48;

board[end\_x][end\_y] = board[start\_x][start\_y];

board[start\_x][start\_y] = 0;

}

bool isValidMove (char\* input, int board[8][8]) {

int start\_y = input[0] - 48;

int start\_x = input[1] - 48;

int end\_y = input[3] - 48;

int end\_x = input[4] - 48;

// out of bounds check

if (start\_x > 7 || start\_x < 0) {

return false;

} else if (start\_y > 7 || start\_y < 0) {

return false;

} else if (end\_x > 7 || end\_x < 0) {

return false;

} else if (end\_y > 7 || end\_y < 0) {

return false;

}

// piece move checks

int\* validMoves[2];

int xcheck = start\_x;

int ycheck = start\_y;

int piecesInWay = 0;

switch (board[start\_x][start\_y]) {

case 0: // moving a blank spot is never valid

return false;

case 1: // King

break;

case 2: // Queen

//printf("Queen Selected\n");

while (true) {

xcheck += (start\_x > end\_x)? -1 : 1;

ycheck += (start\_y > end\_y)? -1 : 1;

if (xcheck < 0 || ycheck < 0 || xcheck > 7 || ycheck > 7) {

break;

}

if (board[xcheck][ycheck] != 0) {

piecesInWay += 1;

}

if (piecesInWay == 2) {

break;

}

if (xcheck == end\_x && ycheck == end\_y) {

return true;

}

}

xcheck = start\_x;

ycheck = start\_y;

piecesInWay = 0;

while (true) {

//printf("Calculating Rook Movement at %d, %d\n", ycheck, xcheck);

if (start\_y == end\_y) {

//printf("Horzontal!\n");

xcheck += (start\_x > end\_x)? -1 : 1;

} else if (start\_x == end\_x) {

//printf("Vertical!\n");

ycheck += (start\_y > end\_y)? -1 : 1;

}

//printf("Now checking %d, %d\n", ycheck, xcheck);

if (xcheck < 0 || ycheck < 0 || xcheck > 7 || ycheck > 7) {

//printf("Out of bounds!\n");

break;

}

if (board[xcheck][ycheck] != 0) {

//printf("Hit Piece: %d\n", board[xcheck][ycheck]);

piecesInWay += 1;

}

if (piecesInWay == 2) {

//printf("Hit Second Piece: %d\n", board[xcheck][ycheck]);

break;

}

if (xcheck == end\_x && ycheck == end\_y) {

return true;

}

}

break;

case 3 : // Bishop

break;

case 4 : // Knight

break;

case 5 : // Rook

break;

case 6 : // Pawn

break;

}

/\*

if (!memberOf(validMoves, end\_x, end\_y)) {

return false;

}

\*/

return true;

}

int main () {

printf("Welcome to Terminal Chess!\n");

printf("Initializing Board...\n") ;

int board [8][8] = { {5, 4, 3, 2, 1, 3, 4, 5}

, {6, 6, 6, 6, 6, 6, 6, 6}

, {0, 0, 0, 0, 0, 0, 0, 0}

, {0, 0, 0, 0, 0, 0, 0, 0}

, {0, 0, 0, 0, 0, 0, 0, 0}

, {0, 0, 0, 0, 0, 0, 0, 0}

, {6, 6, 6, 6, 6, 6, 6, 6}

, {5, 4, 3, 2, 1, 3, 4, 5}

} ;

char buf[100];

bool whitesMove = true;

printf("Board Initialized!\n") ;

// showBoard(board);

printf("Enter `q` to quit at any time.\n");

printf("Moves are entered as co-ordinate pairs, such as \"13-33\" \n");

do {

bool flag = false;

showBoard(board);

do {

if (flag) {

printf("Move Invalid! Try again!");

}

if (whitesMove) {

printf("White to move.\n");

} else {

printf("Black to move.\n");

}

printf("♔ >> ");

scanf("%s", buf);

flag = true;

} while (!isValidMove(buf, board) && buf[0] != 'q' && buf[0] != 'Q');

makeMove(buf, board);

whitesMove = !whitesMove;

} while (buf[0] != 'q' && buf[0] != 'Q') ;

printf("Terminating...\n") ;

}

**FINAL CHESS.C (RIGHT ONE) [from trial.c]**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

// ♔ ♕ ♖ ♗ ♘ ♙

/\* ┌ └ ┐ ┘

├ ┤ ┬ ┴

─ │ ┼

might need these...

╟ ╢ ╧ ╤

\*/

void showBoard (int board[8][8]) {

printf("\n\n 0 1 2 3 4 5 6 7");

printf("\n ╔═╤═╤═╤═╤═╤═╤═╤═╗\n");

for (int i = 0; i < 8; i++) {

printf("%d║",i);

bool flag = true;

for (int j = 0; j < 8; j++) {

if (!flag) {

printf("│");

}

flag = false;

// printf("");

if (board[i][j] == 0) {

if ((i + j) % 2 == 0) {

printf(" ");

} else {

printf("▒");

}

} else if (board[i][j] == 1) {

printf("♔");

} else if (board[i][j] == 2) {

printf("♕");

} else if (board[i][j] == 3) {

printf("♗");

} else if (board[i][j] == 4) {

printf("♘");

} else if (board[i][j] == 5) {

printf("♖");

} else if (board[i][j] == 6) {

printf("♙");

}

}

printf("║\n") ;

if (i < 7) {

printf(" ╟─┼─┼─┼─┼─┼─┼─┼─╢\n");

}

}

printf(" ╚═╧═╧═╧═╧═╧═╧═╧═╝\n");

}

void makeMove (char\* input, int board[8][8]) {

int start\_y = input[0] - 48;

int start\_x = input[1] - 48;

int end\_y = input[3] - 48;

int end\_x = input[4] - 48;

board[end\_x][end\_y] = board[start\_x][start\_y];

board[start\_x][start\_y] = 0;

}

bool isValidMove (char\* input, int board[8][8]) {

int start\_y = input[0] - 48;

int start\_x = input[1] - 48;

int end\_y = input[3] - 48;

int end\_x = input[4] - 48;

// out of bounds check

if (start\_x > 7 || start\_x < 0) {

return false;

} else if (start\_y > 7 || start\_y < 0) {

return false;

} else if (end\_x > 7 || end\_x < 0) {

return false;

} else if (end\_y > 7 || end\_y < 0) {

return false;

}

// piece move checks

int\* validMoves[2];

int xcheck = start\_x;

int ycheck = start\_y;

int piecesInWay = 0;

switch (board[start\_x][start\_y]) {

case 0: // moving a blank spot is never valid

return false;

case 1: // King

if (abs(start\_x - end\_x) <= 1 && abs(start\_y - end\_y) <= 1) {

return true;

}

break;

case 2: // Queen

//printf("Queen Selected\n");

while (true) {

xcheck += (start\_x > end\_x)? -1 : 1;

ycheck += (start\_y > end\_y)? -1 : 1;

if (xcheck < 0 || ycheck < 0 || xcheck > 7 || ycheck > 7) {

break;

}

if (board[xcheck][ycheck] != 0) {

piecesInWay += 1;

}

if (piecesInWay == 2) {

break;

}

if (xcheck == end\_x && ycheck == end\_y) {

return true;

}

}

xcheck = start\_x;

ycheck = start\_y;

piecesInWay = 0;

while (true) {

//printf("Calculating Rook Movement at %d, %d\n", ycheck, xcheck);

if (start\_y == end\_y) {

//printf("Horzontal!\n");

xcheck += (start\_x > end\_x)? -1 : 1;

} else if (start\_x == end\_x) {

//printf("Vertical!\n");

ycheck += (start\_y > end\_y)? -1 : 1;

}

//printf("Now checking %d, %d\n", ycheck, xcheck);

if (xcheck < 0 || ycheck < 0 || xcheck > 7 || ycheck > 7) {

//printf("Out of bounds!\n");

break;

}

if (board[xcheck][ycheck] != 0) {

//printf("Hit Piece: %d\n", board[xcheck][ycheck]);

piecesInWay += 1;

}

if (piecesInWay == 2) {

//printf("Hit Second Piece: %d\n", board[xcheck][ycheck]);

break;

}

if (xcheck == end\_x && ycheck == end\_y) {

return true;

}

}

break;

case 3 : // Bishop

while (true) {

xcheck += (start\_x > end\_x)? -1 : 1;

ycheck += (start\_y > end\_y)? -1 : 1;

if (xcheck < 0 || ycheck < 0 || xcheck > 7 || ycheck > 7) {

break;

}

if (board[xcheck][ycheck] != 0) {

piecesInWay += 1;

}

if (piecesInWay == 2) {

break;

}

if (xcheck == end\_x && ycheck == end\_y) {

return true;

}

}

break;

case 4 : // Knight

if (abs(start\_x - end\_x) == 1) {

if (abs(start\_y - end\_y) == 2) {

return true;

}

} else if (abs(start\_x - end\_x) == 2) {

if (abs(start\_y - end\_y) == 1) {

return true;

}

}

break;

case 5 : // Rook

while (true) {

if (start\_y == end\_y) {

xcheck += (start\_x > end\_x)? -1 : 1;

} else if (start\_x == end\_x) {

ycheck += (start\_y > end\_y)? -1 : 1;

} else {

return false;

}

if (xcheck < 0 || ycheck < 0 || xcheck > 7 || ycheck > 7) {

break;

}

if (board[xcheck][ycheck] != 0) {

piecesInWay += 1;

}

if (piecesInWay == 2) {

break;

}

if (xcheck == end\_x && ycheck == end\_y) {

return true;

}

}

break;

case 6 : // Pawn

//printf("Pawn Selected\n");

//printf("Moving in column %d \n", y1);

//printf("y1-y2=%d\n", x1-x2);

if (start\_x - end\_x == 1 && start\_y == end\_y) { // forward one

printf("Detected forward move by 1\n");

if (board[end\_x][end\_y] == 0) {

return true;

}

} else if (start\_x - end\_x == 2 && start\_y == end\_y) { // forward two

// printf("Detected forward move by 2\n");

if (board[end\_x][end\_y] == 0 && board[end\_x+1][end\_y] == 0 && start\_x == 6) {

return true;

}

} else if (abs(start\_y - end\_y) == 1 && start\_x - end\_x == 1) { // capture left or right

//printf("Detected Capture\n");

if (board[end\_x][end\_y] != 0) {

return true;

}

}

break;

}

}

int main () {

printf("Welcome to Terminal Chess!\n");

printf("Initializing Board...\n") ;

int board [8][8] = { {5, 4, 3, 2, 1, 3, 4, 5}

, {6, 6, 6, 6, 6, 6, 6, 6}

, {0, 0, 0, 0, 0, 0, 0, 0}

, {0, 0, 0, 0, 0, 0, 0, 0}

, {0, 0, 0, 0, 0, 0, 0, 0}

, {0, 0, 0, 0, 0, 0, 0, 0}

, {6, 6, 6, 6, 6, 6, 6, 6}

, {5, 4, 3, 2, 1, 3, 4, 5}

} ;

char buf[100];

bool whitesMove = true;

printf("Board Initialized!\n") ;

// showBoard(board);

printf("Enter `q` to quit at any time.\n");

printf("Moves are entered as co-ordinate pairs, such as \"13-33\" \n");

do {

bool flag = false;

showBoard(board);

do {

if (flag) {

printf("Move Invalid! Try again!");

}

if (whitesMove) {

printf("White to move.\n");

} else {

printf("Black to move.\n");

}

printf("♔ >> ");

scanf("%s", buf);

flag = true;

} while (!isValidMove(buf, board) && buf[0] != 'q' && buf[0] != 'Q');

makeMove(buf, board);

whitesMove = !whitesMove;

} while (buf[0] != 'q' && buf[0] != 'Q') ;

printf("Terminating...\n") ;

}