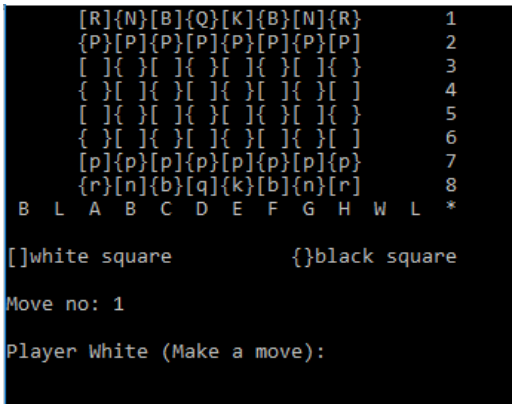
Chess !!

The following report explains the functions used in the making of the game



Note that pdf of the flow charts are attached to this folder, and the explanation of the game and all the assumption will be shown in the game right before starting

so lets get started !!

At first we need to indicate the principle on which the game's board is based, which is a three dimensional array.

Why is that? Well, the first two dimensions are for the board's shape, while the third one is foe the undo and redo, but we'll get to that in details later.

Printboard();

The printboard() guarantee that the board printed for the user is constant no matter what changes happens in the core of the game.

It consists of multiple for loops and if statements and a variable named flag that starts with initial value of 1.

The role of the flag is to determine the nature of the first block of the board and it changes constantly so that each row begins with a different type of block (as white and black squares).

Afterward, by looping on all the squares of the row all what the program has to do is to see the last square's type ([] or {}) so that it becomes the other one and so one.

C_toboard();

This function is to transform the input of the user for the columns coordinates into numbers that represent a certain location in the array with a switch statement.

R_toboard();

It does the same as the previous on but for the rows, because the numbers that the user can see on the right side of the board exceed the row number by 1.

teamW() & teamb() ;

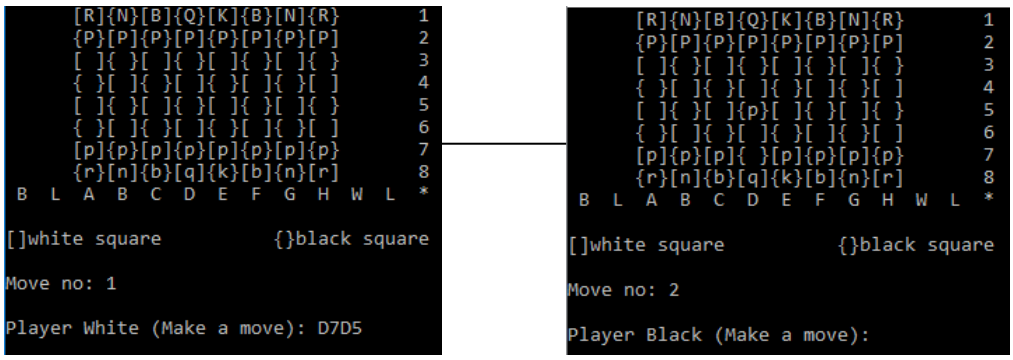
These two functions simply check if a certain piece is a black or a white piece, so if the piece is {p,n,b,r,q,k} it's a white one and teamw(); will return a true value indicating this ,while if it is {P,N,B,R,Q,K} it's a black one and teamb(); is the one that going to be returning the true value.

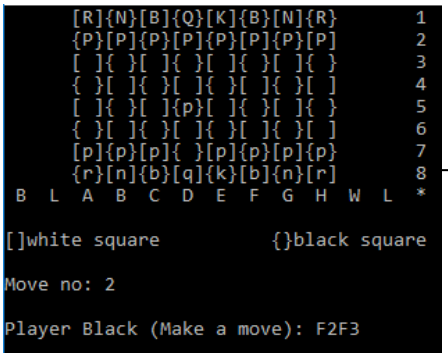
Notice that if a black piece is checked by teamw(); the returned value will be false and vise versa.

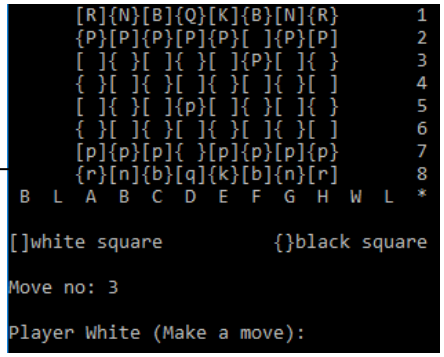
Pmove(); & pmove();

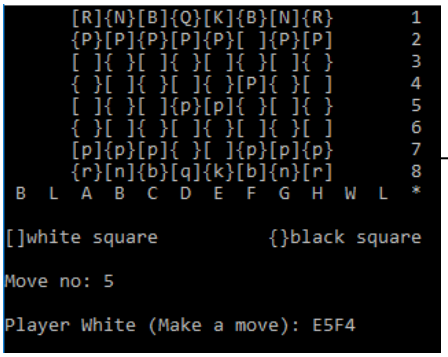
These two functions indicates how the pawn should move ,the pawn is the only piece that has two motion functions one for the white and one for the black because it moves only in the direction of the enemy ,which means upward for the white and downward for the black.

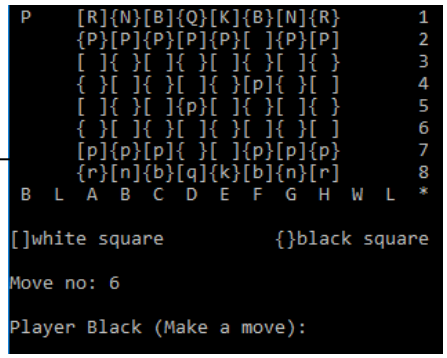
This function take a certain a move from the user and check if a regular pawn would move this way (example: moving to the front by 1 step or 2 if it is his first play with this pawn , or eating an enemy by moving diagonally) so if yes it returns a true value



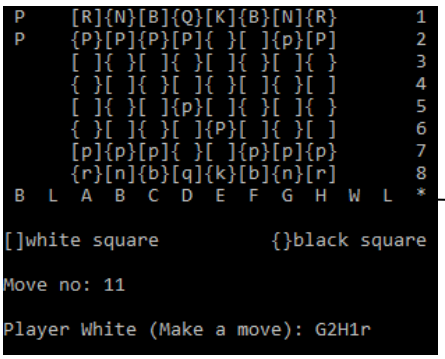


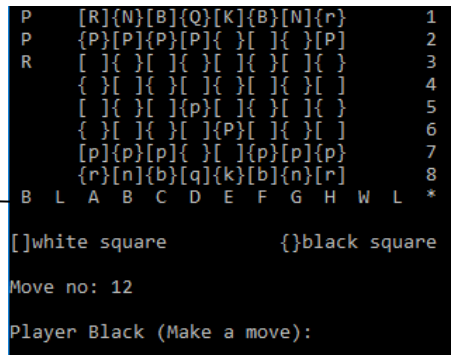




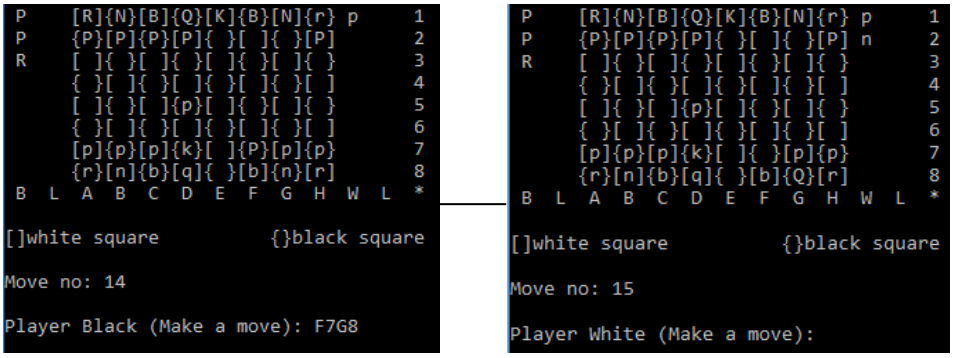


The second part (or shall I say first because it's written first) is checking for promotion ,so that if the user reaches the last lines of the enemy's ,he can promote his/her pawn to a higher character. The user enters the coordinates of his move as normal followed by the piece he wants to promote his pawn to.



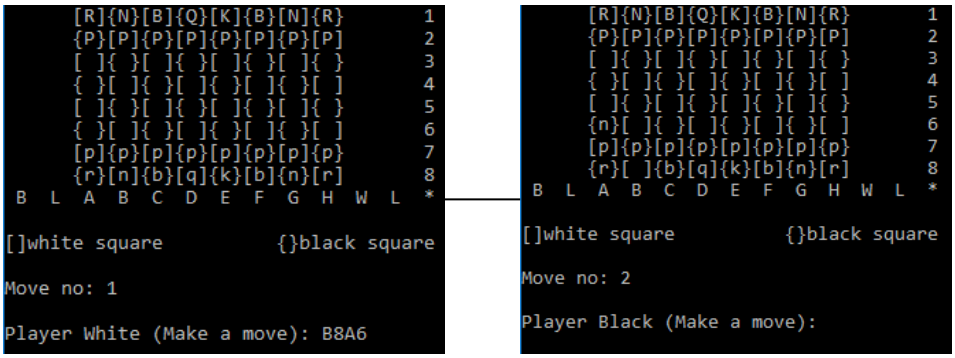


Note: If the player didn’t chose the piece he wants to promote his pawn to, the pawn will promote to a queen by default.



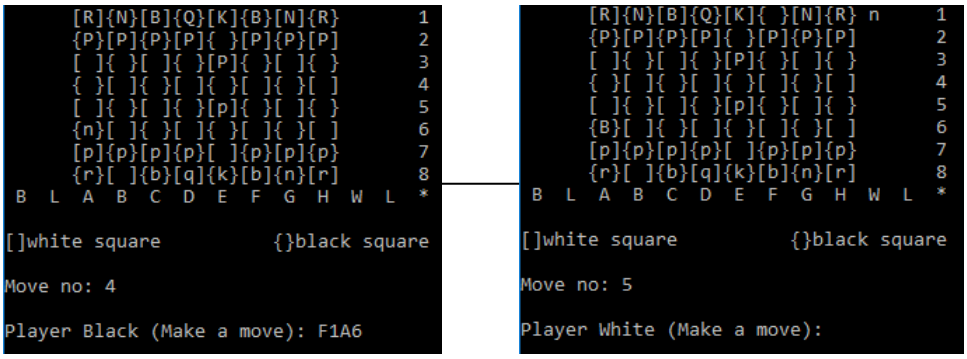
nmove();

This function ensures that the knight moves as it should which means that either its x or y coordinates has a difference of two while the other is one, and it returns a true value if it does.



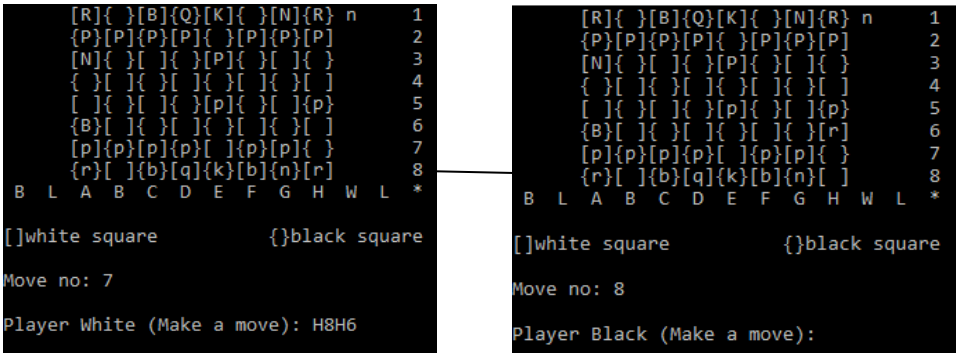
bmove();

This function ensures that the bishop moves as it should which means that both its x and y coordinates has equivalent differences ,while also checking that there are no pieces blocking the movement .



rmove();

This function ensure that the rooks moves as it should which means that either its x and y coordinate is constant while the other changes ,while also checking that there are no pieces blocking the movement .

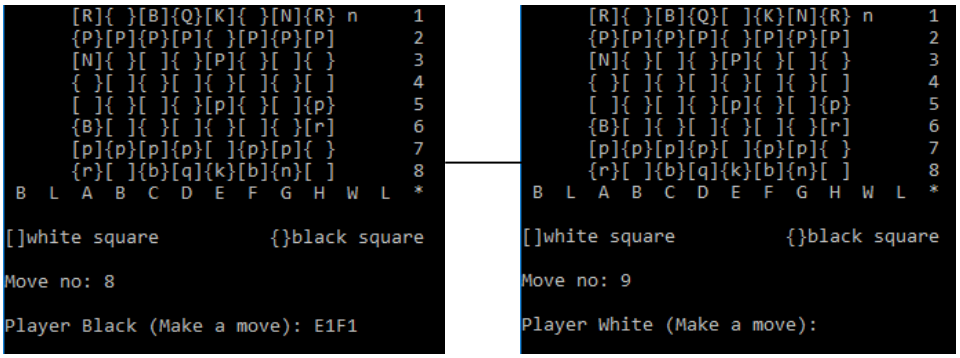


qmove();

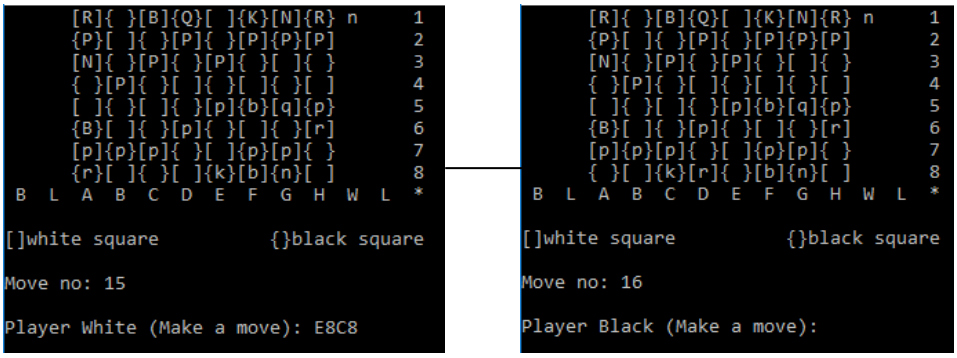
This function ensure that the queen moves as it should, It is a combination between bishop and rook motions, while also checking that there are no pieces blocking the movement.

kmove();

This function ensure that the king moves as it should, its only task is to ensure that the difference between x or y coordinates or maybe both is less than or equal to one.



Also, one of the most well-known tactical moves for the king in chess is the Castling move. As long as the king and one of the rooks haven't moved yet, the king could move two squares towards the rook and the rook moves to the square over which the king crossed. In the castling move there must be no pieces intercepting the moves of both pieces.



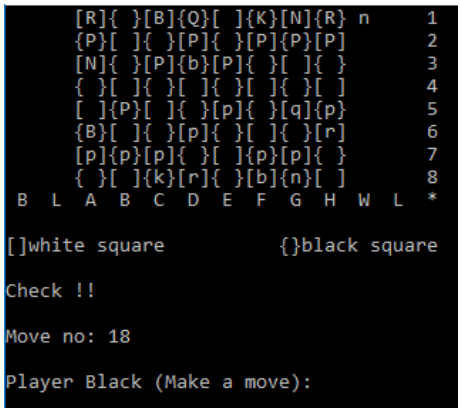
Valid();

This function is the mother of all previous motion functions, it takes an input from the user indicating a certain piece and a certain location desired for this piece to go to, and it filters it. It checks what type of piece is located in the input then send the desired location to this piece's function to check if it is a proper move or not.

After that, job isn't done yet. If the move is proper that's good but it needs to check if there is a check on the king, and does this move exclude this threat for the king or does it cause one. So that in the end a player can only do a correct move that doesn't cause a check for his king.

Check();

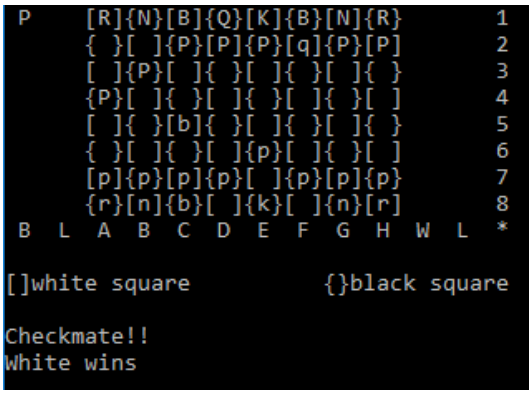
The check function is pretty simple, it takes the location of the king and start looping over the whole board to see if there is any valid move for any piece to this location, if there is well that's a check but if there isn't than that's nice your king is safe.

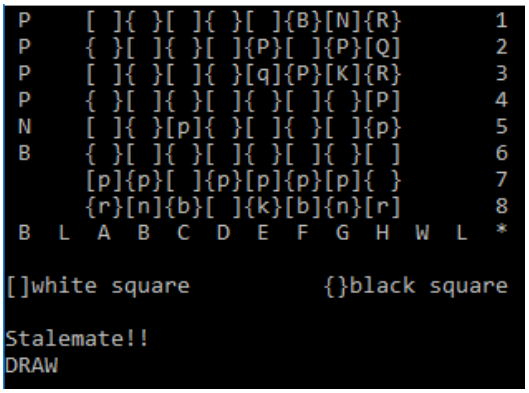


"In this case, the white bishop in d3 has a valid move to the black king in f1. So it's a check:

Endgame();

You can relate to its functionality from her name, this function checks for possible moves for the player on the whole board ,it loops and each time it finds one of your pieces on the board it start looping to find any valid move for this piece. If no valid moves are found then it sees if there is a check on your king for the game to end in a checkmate and a victory for your opponent, and if not then that's a draw and a stalemate is declared.





Move();

If the valid function returns a positive, then it's time to move the piece and here comes this function's turn. First of all it checks if the piece is a king or not, because if it is we'll need the new location saved in order to use it in the check function. After that it checks if the motion contain a pawn that's going to be promoted so that it changes it to the desired piece, after that it simply changes the locations of the pieces on the board indicating the motion, and if any piece was eaten during the process it excludes it from the board and puts it in the two columns of the lost pieces if it was black or white.

A secondary function here is to count the moves of both kings and each one of the four rooks. So that at any time, if a player wants to castle his king, both of the desired king and the desired rook should have no previous moves.

Save();

The save function opens a save text file and store the turn of the last player and the board with all the moves and lost pieces so that it could be loaded at any time.

Load();

The load function opens the save text file and start taking row by row all the elements and storing it in the board until the program reaches the end of the file, and so the board is updated with the previous saved one.

Undo();

As long as the move number is greater than one, a player can keep undoing the moves till the first turn. This is why the board is actually three dimensional, so that the third dimension stores the active move in which the game is currently at, by typing UNDO the player orders the game to decrease the active moves variable and so go back in the previous layer of the game. It also searches for the king's place in this previous turn so that it changes its position in the board while undoing the last move.

Redo();

Similar to the last mentioned function, this one displays the next layer of the board array if there any available layers to be shown. It also searches for the king's place and changes its position in the board.

Main();

The main function organizes all the function and use them to create the game as it looks like, first of all it loads the READ ME.txt which contains instructions about the game and all the assumptions used in it, next it enters the players in the game and start taking inputs from them and check if the input is a usable sentence such as SAVE,LOAD,UNDO,REDO so that the game execute it.

After taking the input and checking if it is an existing location on the board it starts calling the valid function, if the move is valid then the move function is called and the game is updated.

It also checks in the beginning of the code for checks or if it is and endgame point. If not, then it takes new inputs and so the game is resumed.

PLEASE NOTE THAT FLOWCHARTS FOR MOST OF THESE FUNCTIONS ARE ATTACHED TO THIS FOLDER AND THAT ALL THE GAME INSTRUCTION ARE LOCATED IN READ ME.TXT AND WILL BE DISPLAYED IN THE BEGINNING OF THE GAME.

HAPPY GAMING

CHESS!!

By Youssef Sherif Nashaat

Ramez Maher Victor