Chess project

Introduction:

First, I'd like to thank Dr.NAGIA for introducing this useful course to us and making it as easy as possible for all of us to understand the course, actually DR.NAGIA has put me on the first step of programming and that's really cool , I and my team mate ashraf also thank engineer AHMED GHAZAL and OMAR SALAH ELDIN for being very helpful to us and explaining some parts of projects , I think that we have benifted too much from this experience .

Second, I'd like to say that the best thing I have benefited from is cooperation between me and my team mate it has been a useful experience .

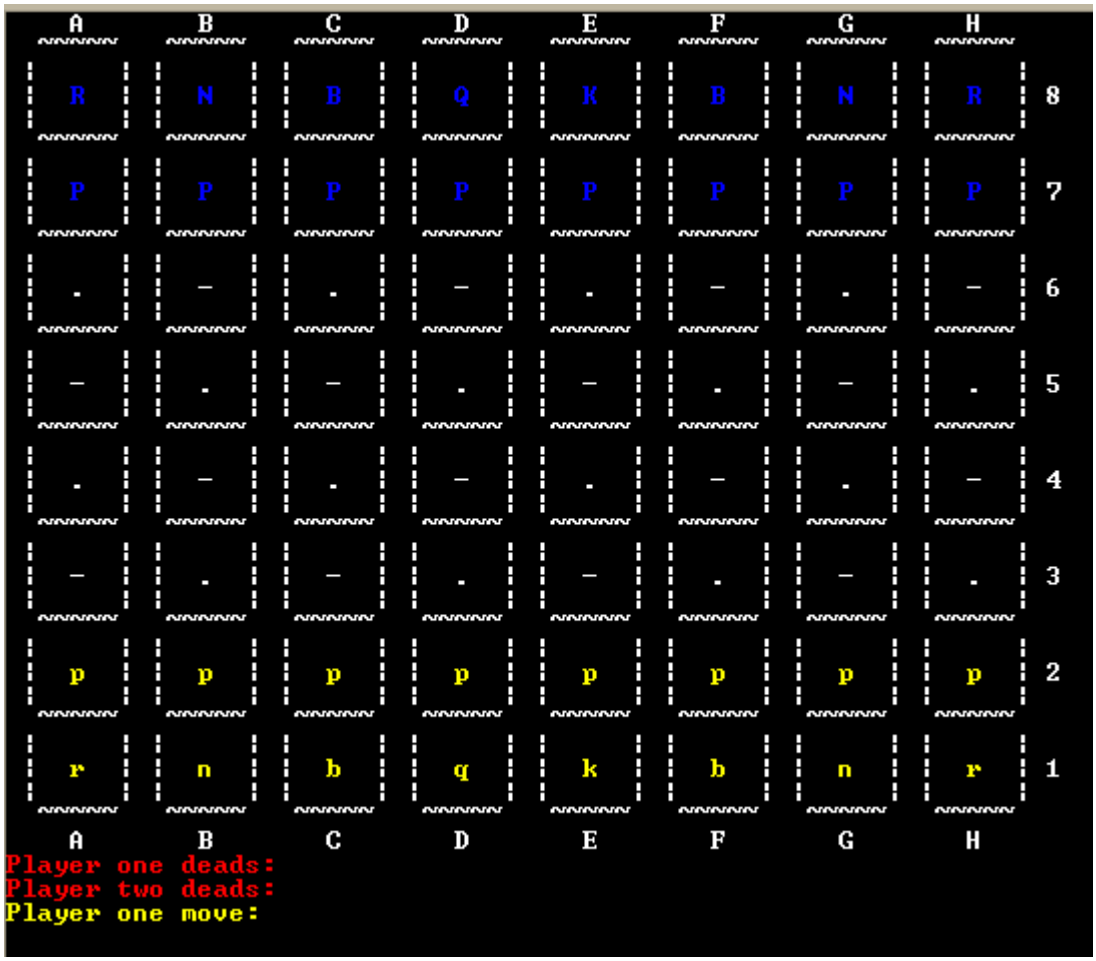
Game description:

It’s a chess game which involves player v.s player mode or player v.s computer where player win when

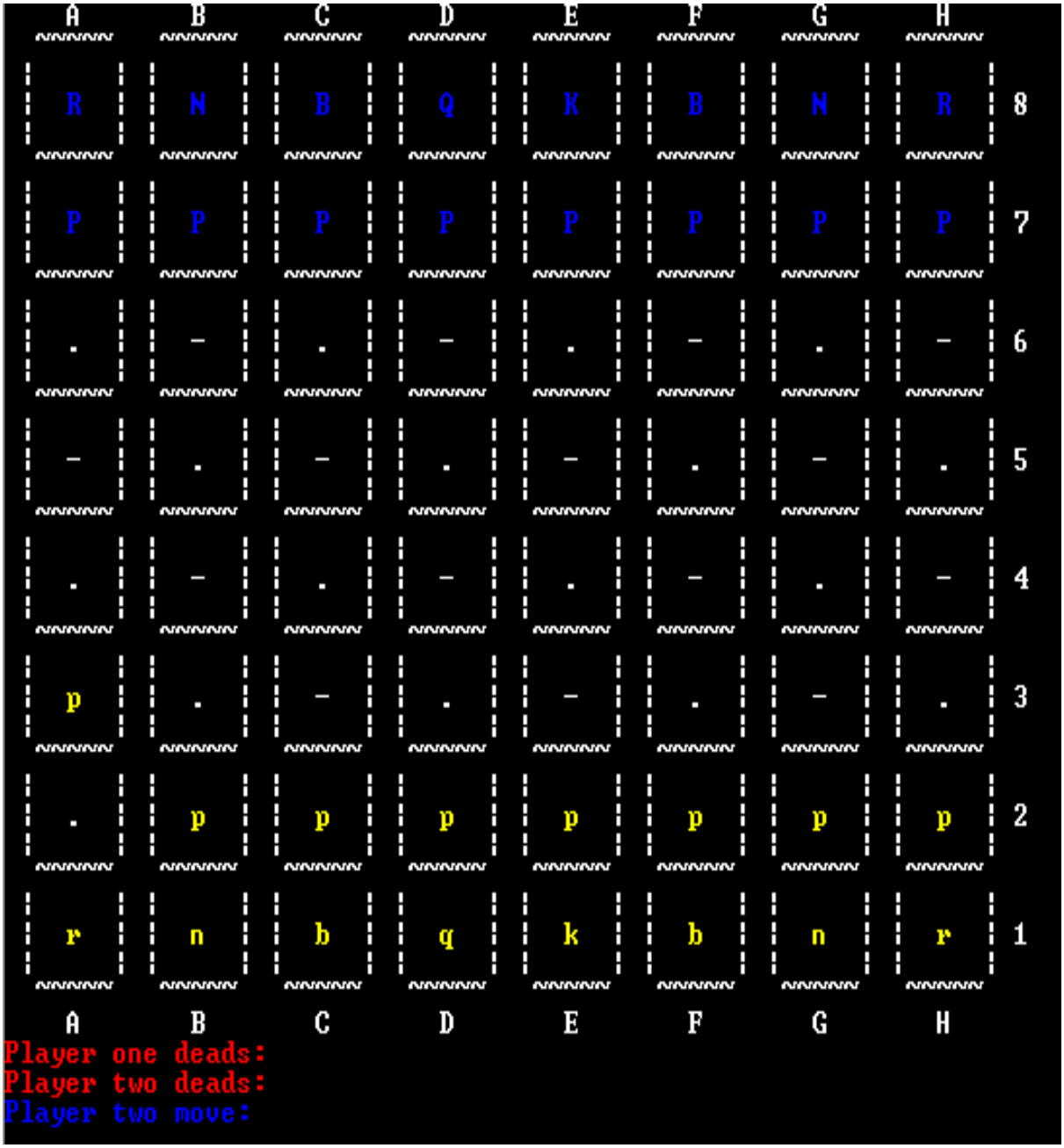
The others players king is check mated and is check mated in any other place or when the others player king is stale mated it provides undo/redo system and also save/load system and this how the program look like :

Program home page:

And this is board structure:

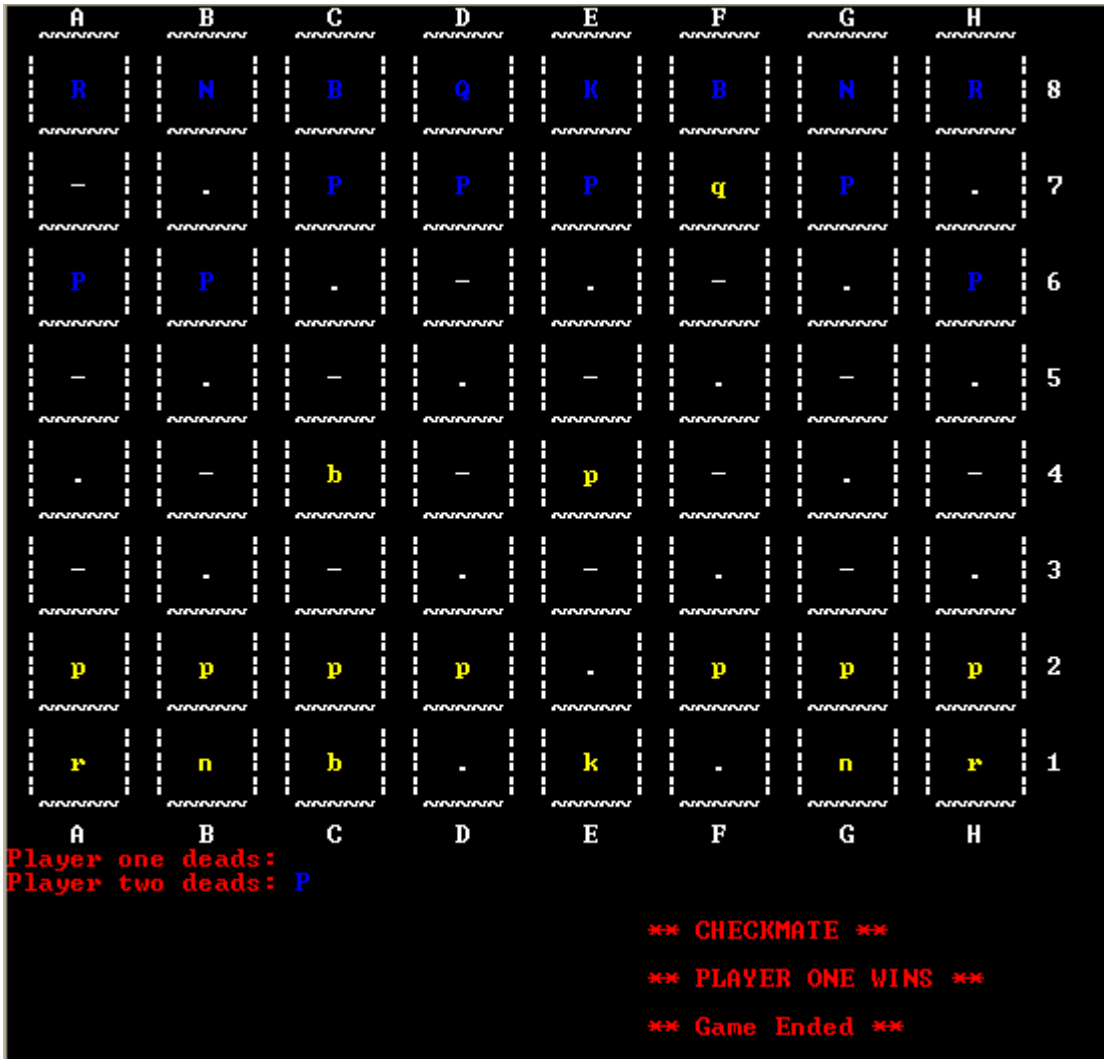


Player1 move:



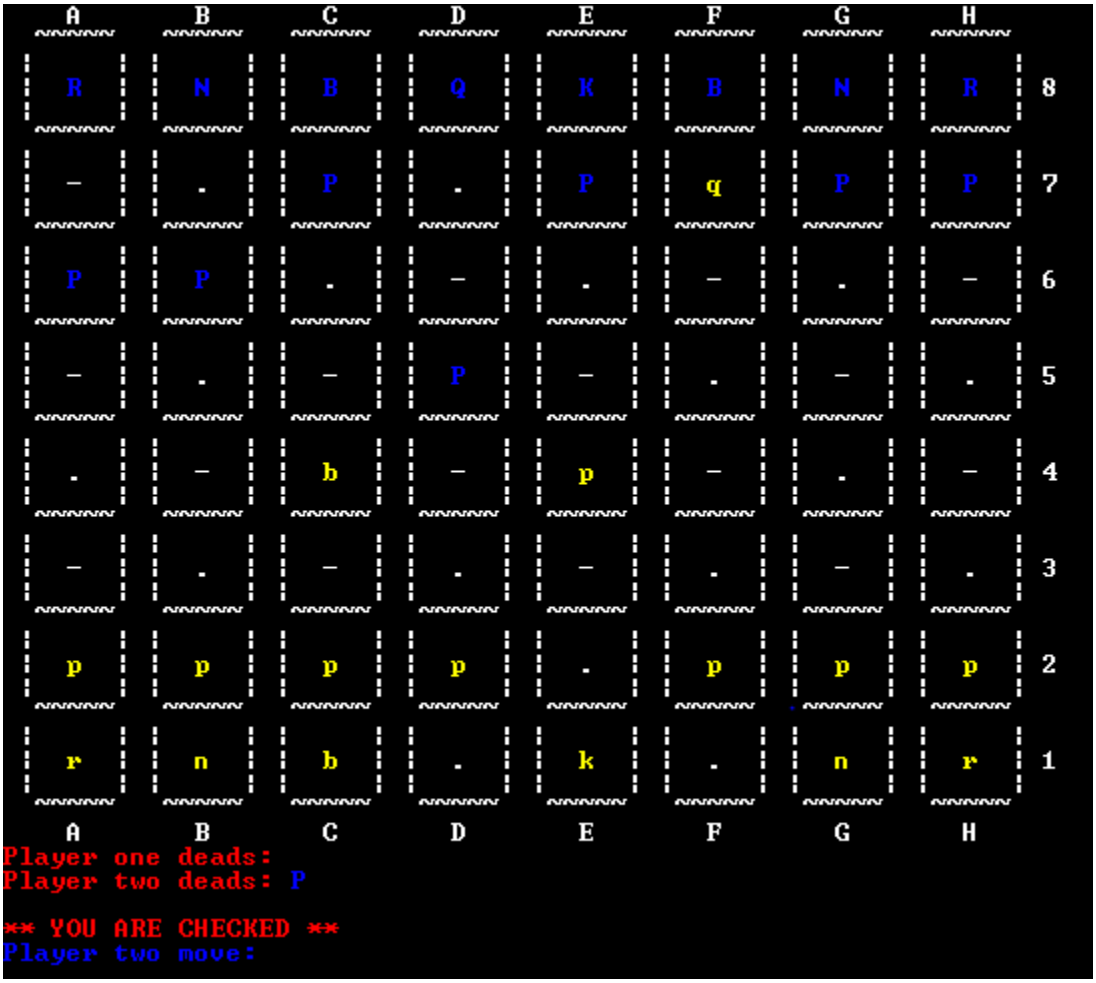
Checkmate:

White first: E2E4, H7H6, D1H5, A7A6, F1C4, B7B6, **H5F7** (Pawn eaten and checkmate)



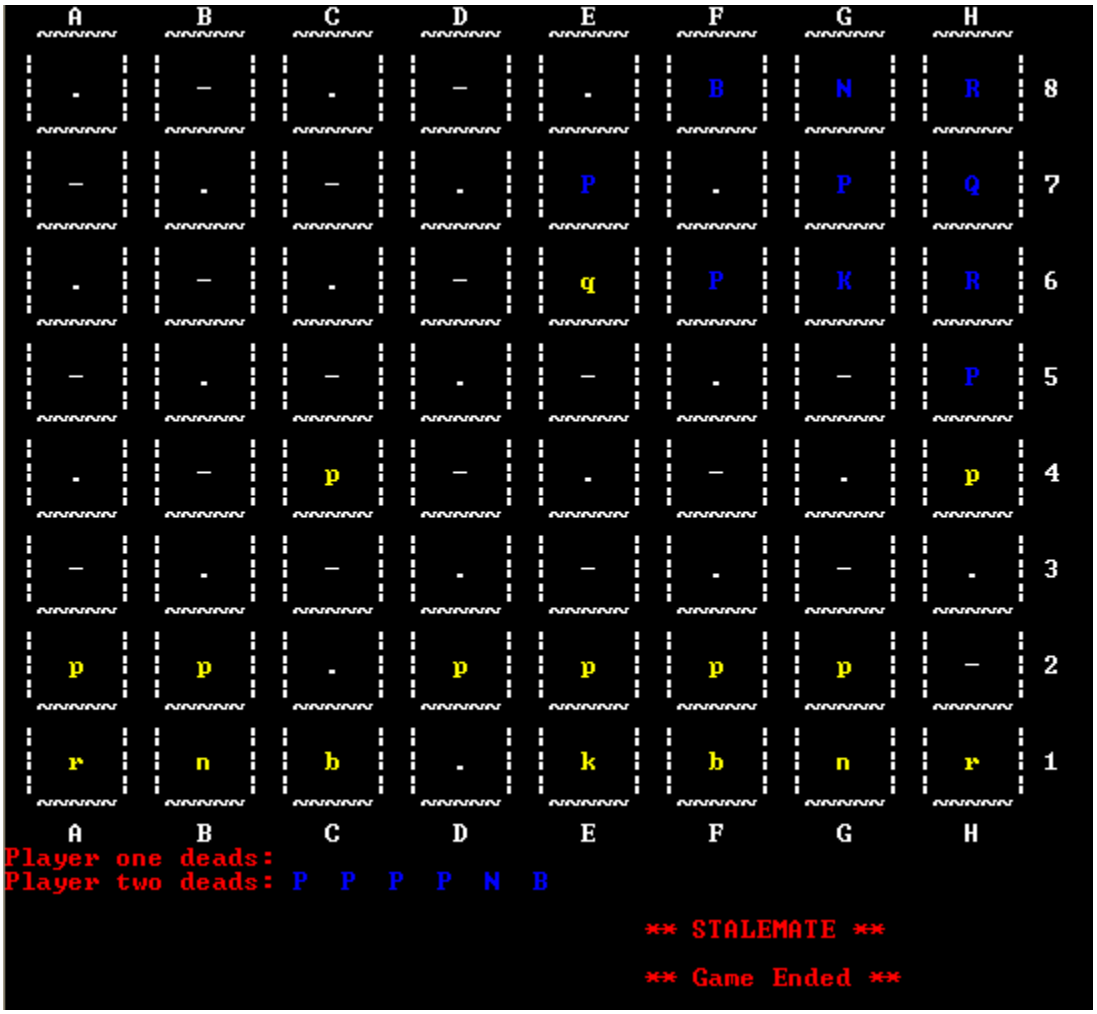
Check:

2- White first: E2E4, D7D5, D1H5, A7A6, F1C4, B7B6, **H5F7** (Check)



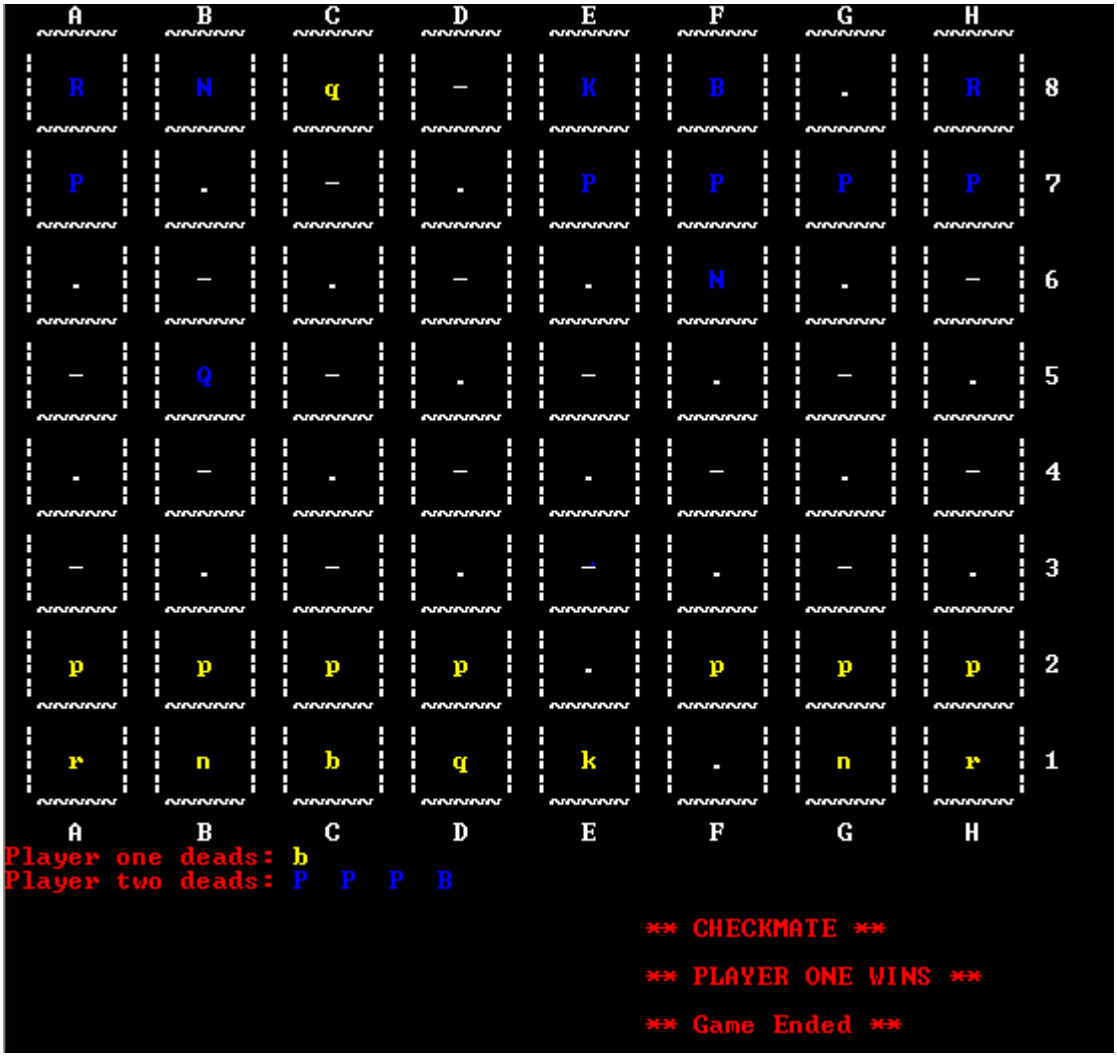
Stalemate

1- White first: C2C4, H7H5, H2H4, A7A5, D1A4, A8A6, A4A5, A6H6, A5C7, F7F6, **C7D7** (Check), E8F7, D7B7, D8D3, B7B8, D3H7, B8C8, F7G6, C8E6



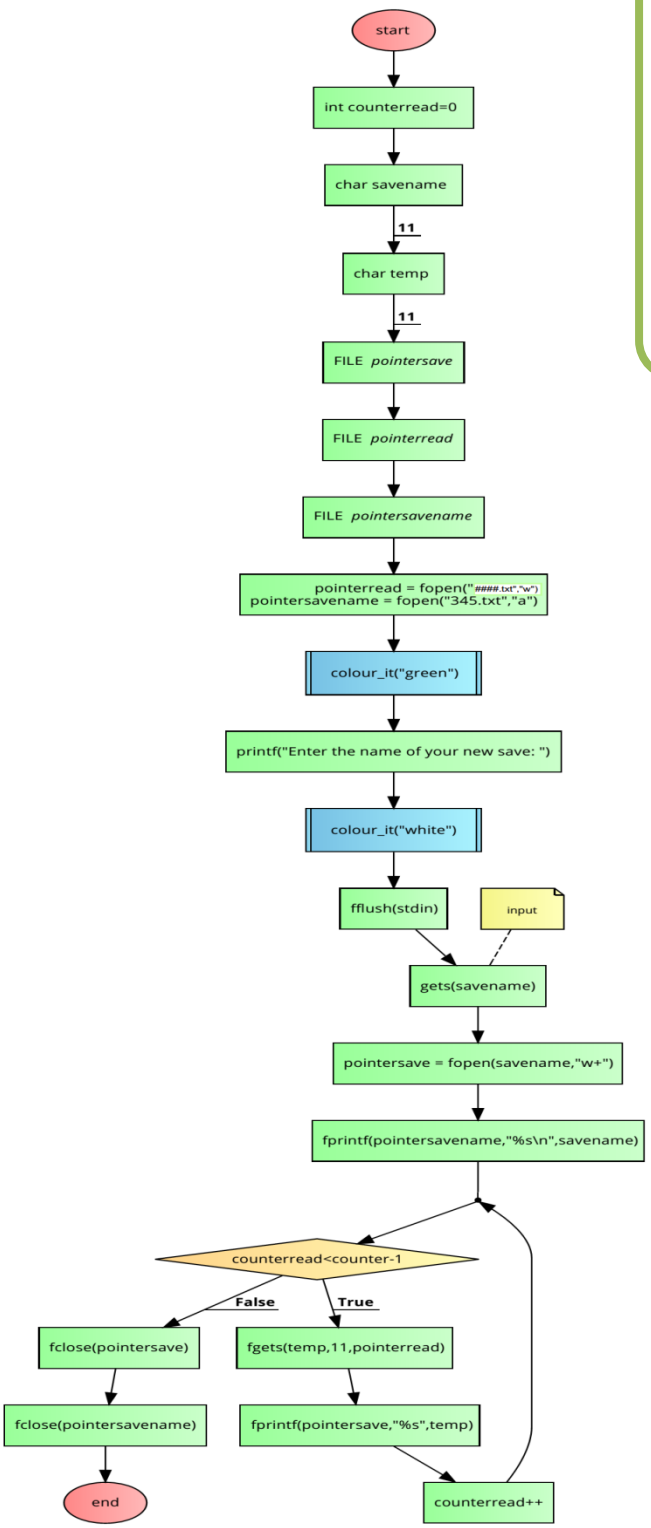
promotion

1- White first: E2E4, D7D5, E4D5, G8F6, **F1B5** (Check), C7C6, D5C6, D8B6, **C6B7** (Check), B6B5, **B7C8** (Promotion to queen and checkmate)



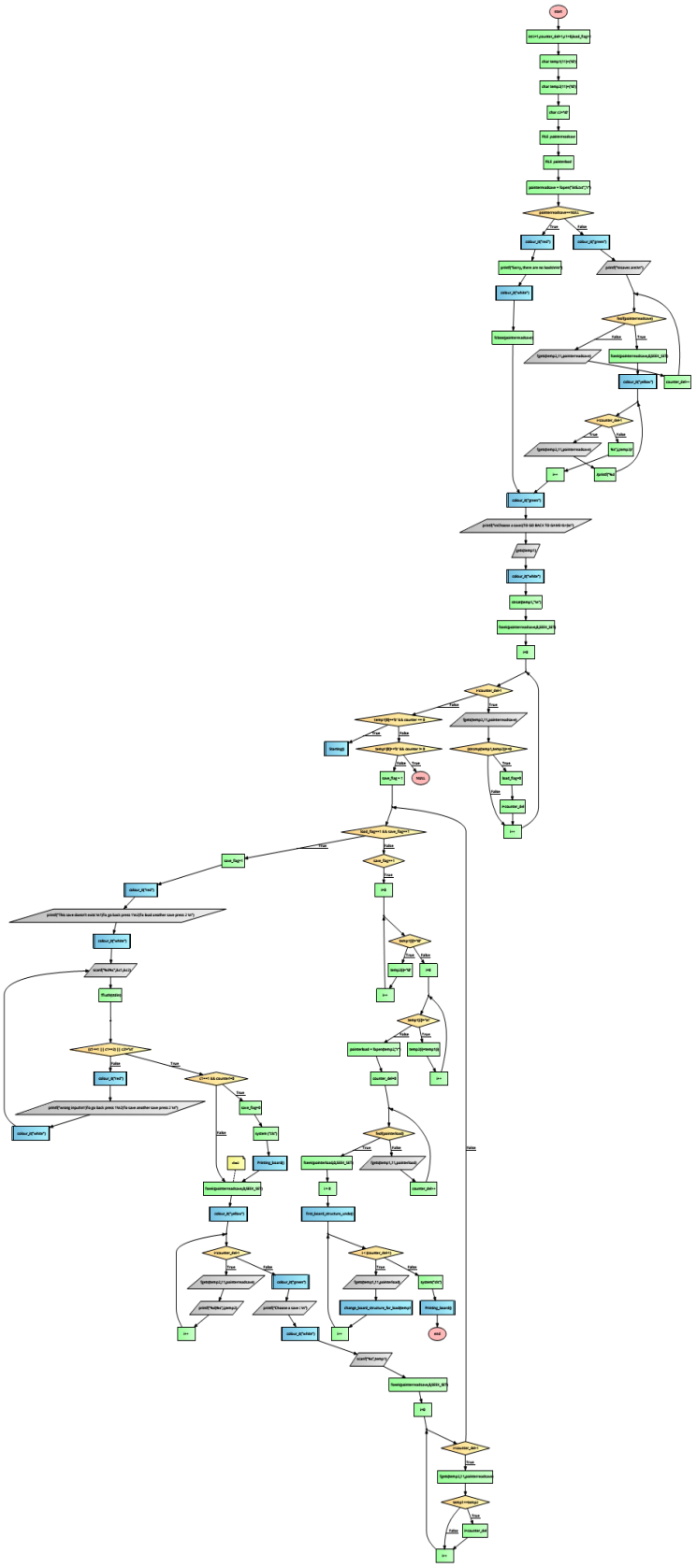
Important flowcharts

save flowchart:



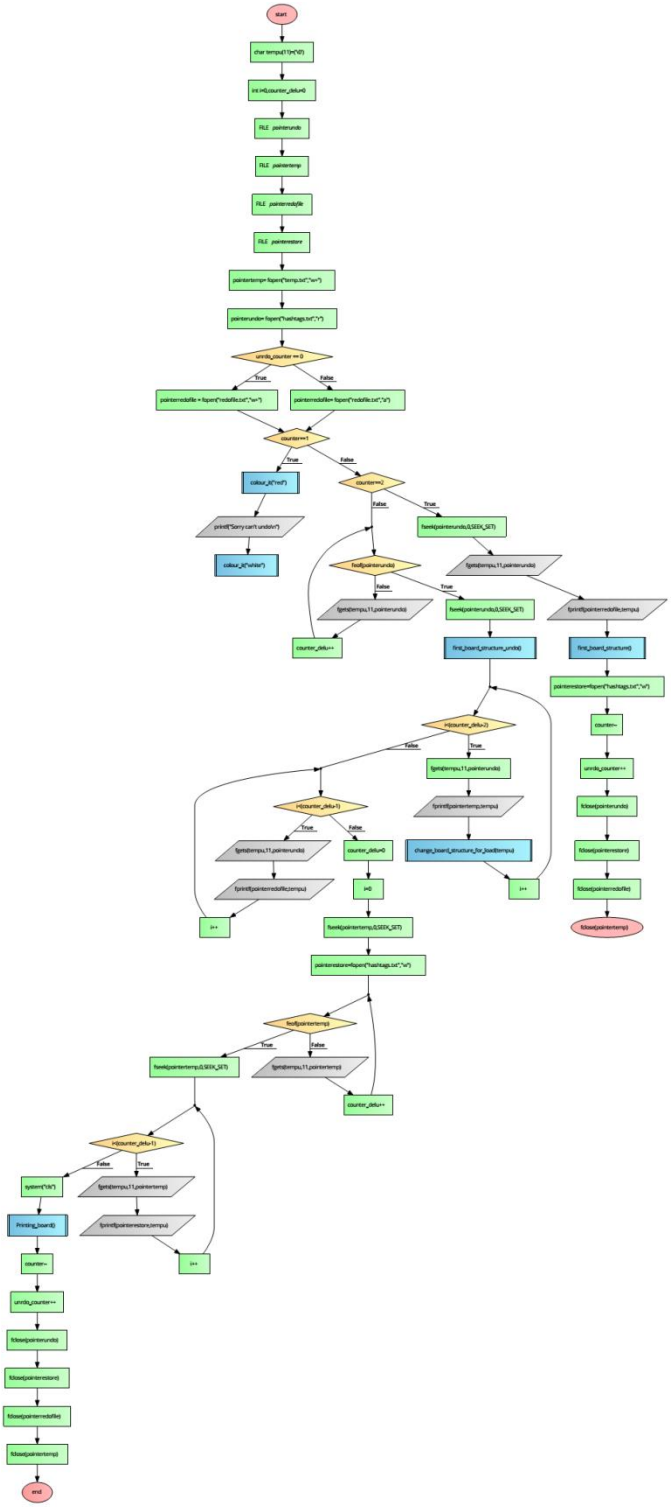
in save system we used file input/output to copy data of the steps of execution out side program in text file where inputs is saved step by step and on loading it will be executed in sequence .

Load inside flow chart:



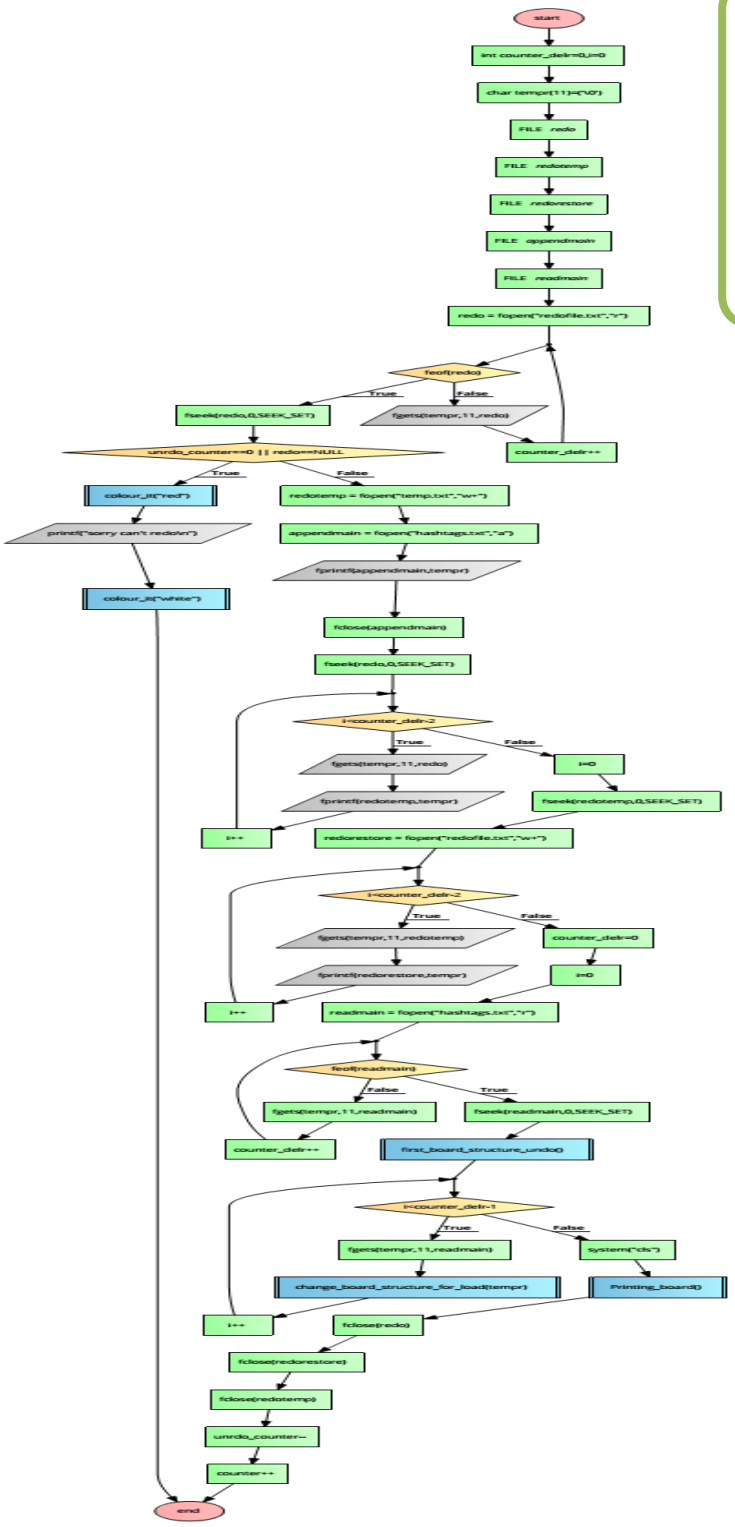
In loading system we only copy data that has been stored in txt file again to the program to be executed step by step until the last move in that save.

Undo flow:



In undo system we use the text file that we save moves in it step by step and execute all moves until the previous move and copy this move to another text file to be used in redo in the future .

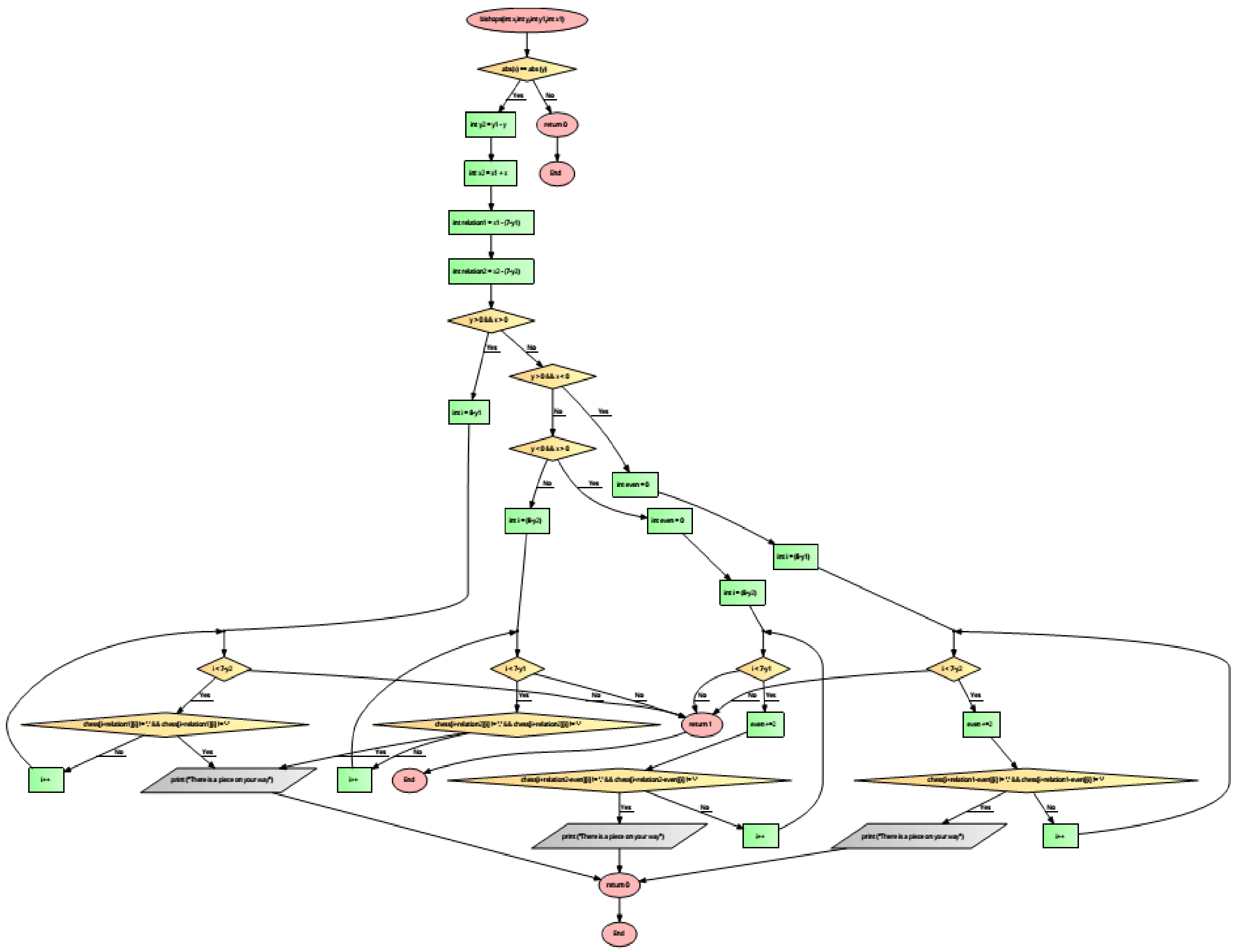
Redo flow:



In redo system we used file that we copied undo moves to it and execute the whole previous moves in addition to the last copied move in that text file.

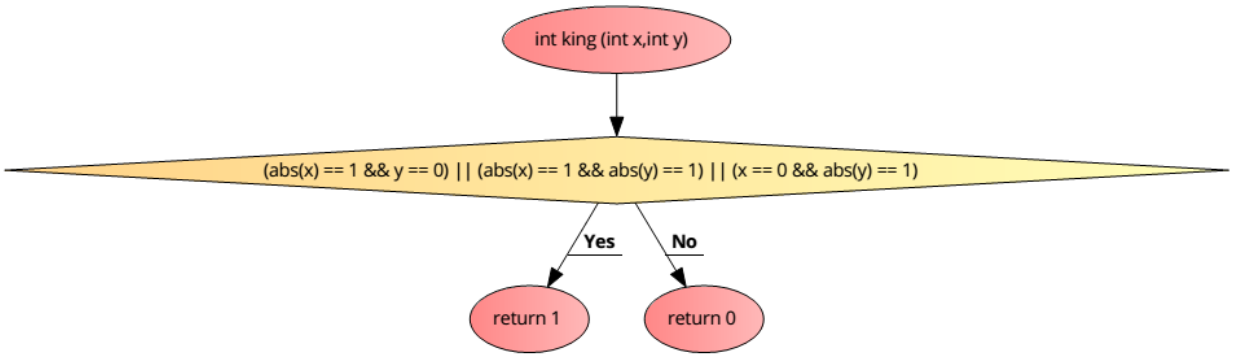
Bishops :

Check the absolute difference between the beginning square and the destination square in X is equal to that in Y and check that there is nothing on your way to the destination square



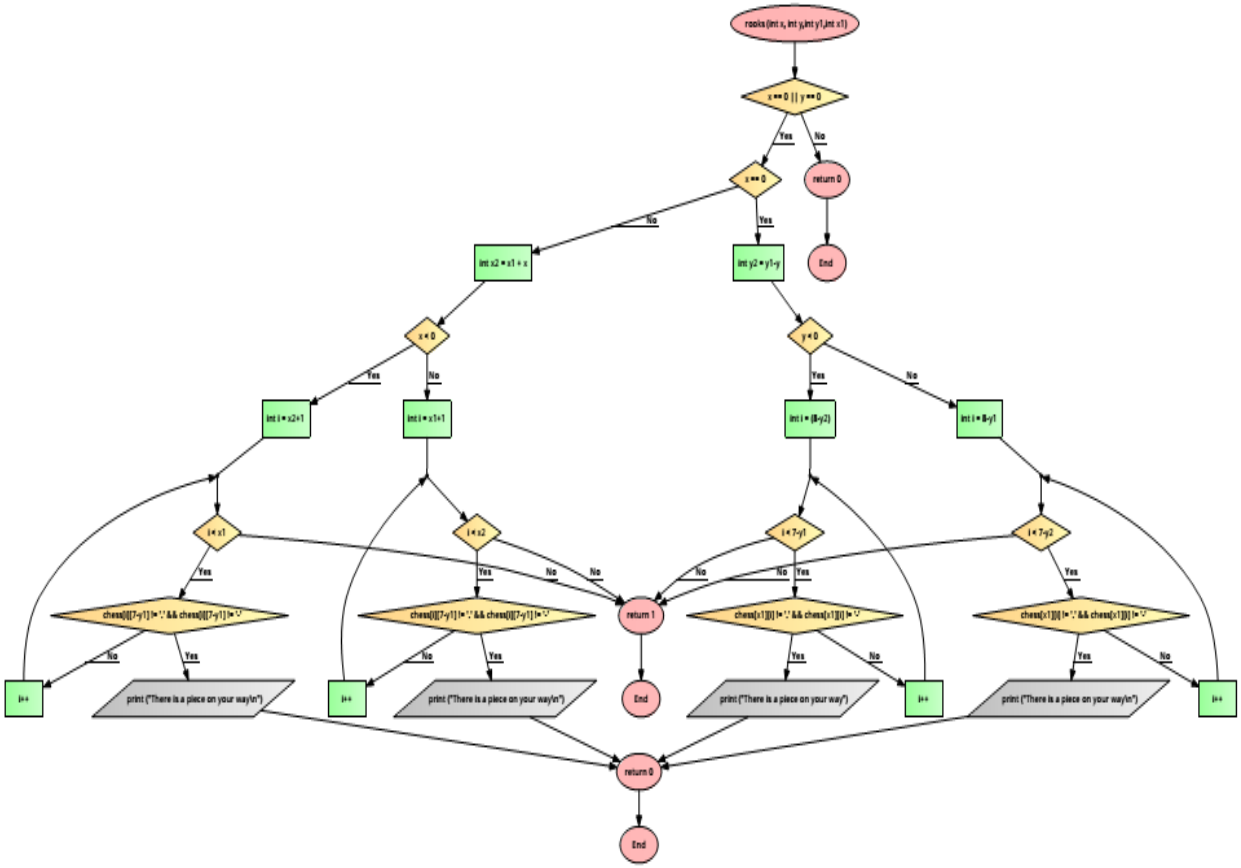
King:

Check the absolute difference between the beginning square and the destination square in X and Y that it is at most 1 square and that it doesn't contain one of its own pieces



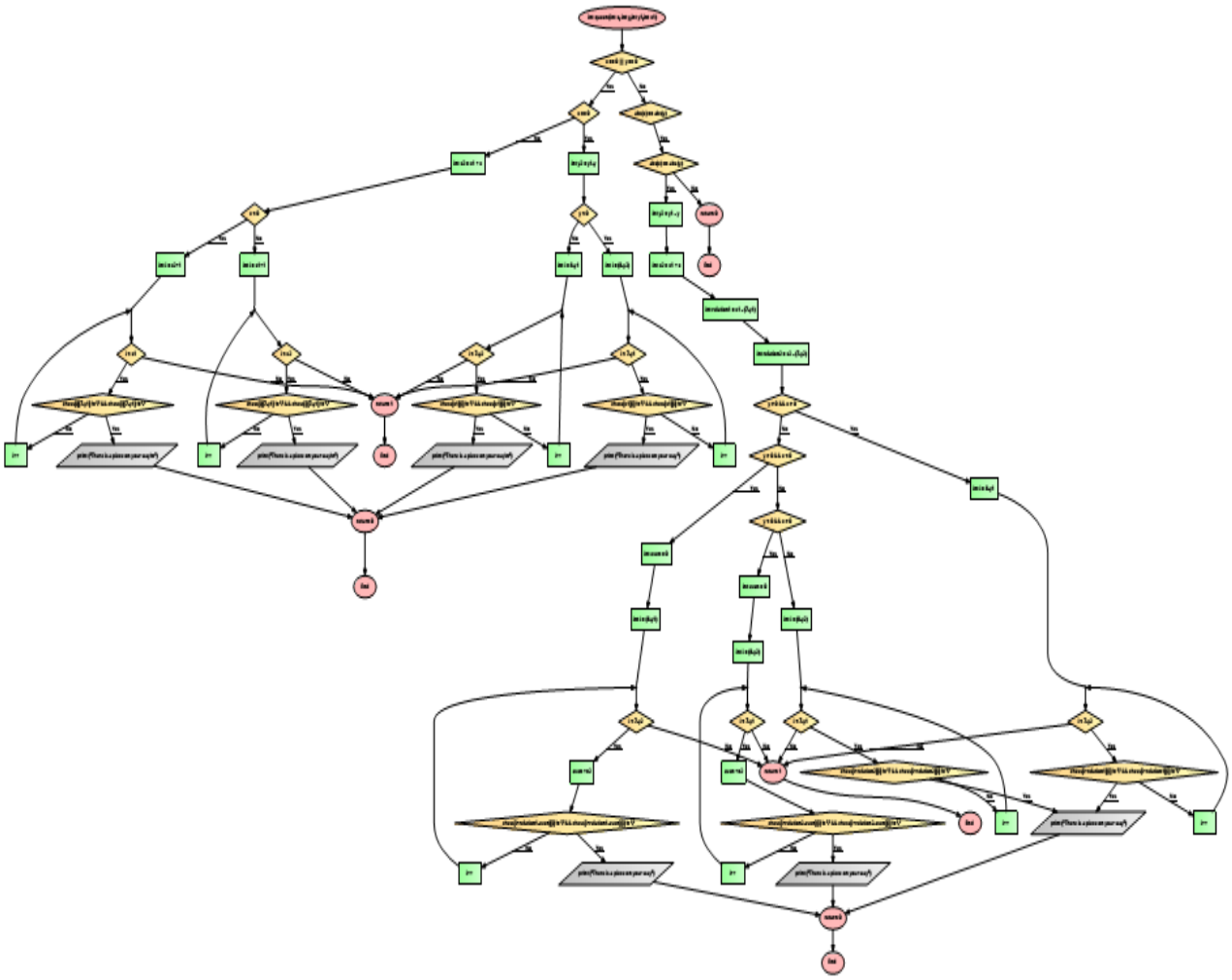
Rooks:

check the difference between the beginning square and the destination square is Zero in either X or Y and check that there is nothing on your way to the destination square.



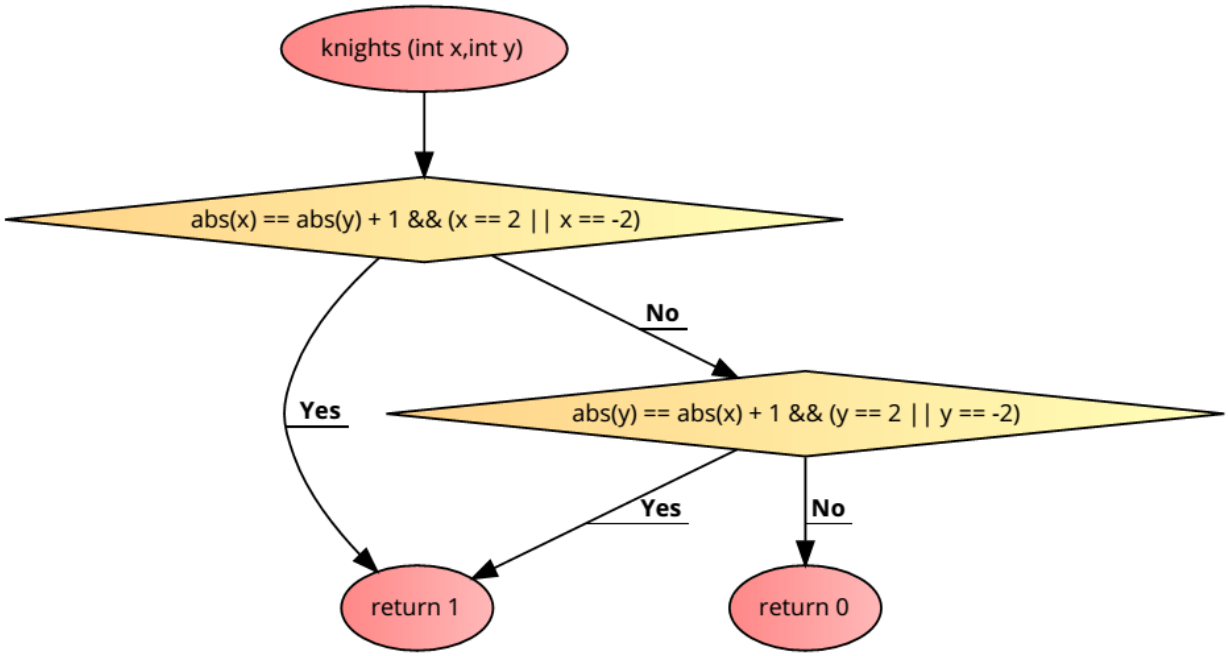
Queen:

it is a combination between Rooks () function and Bishops () function.



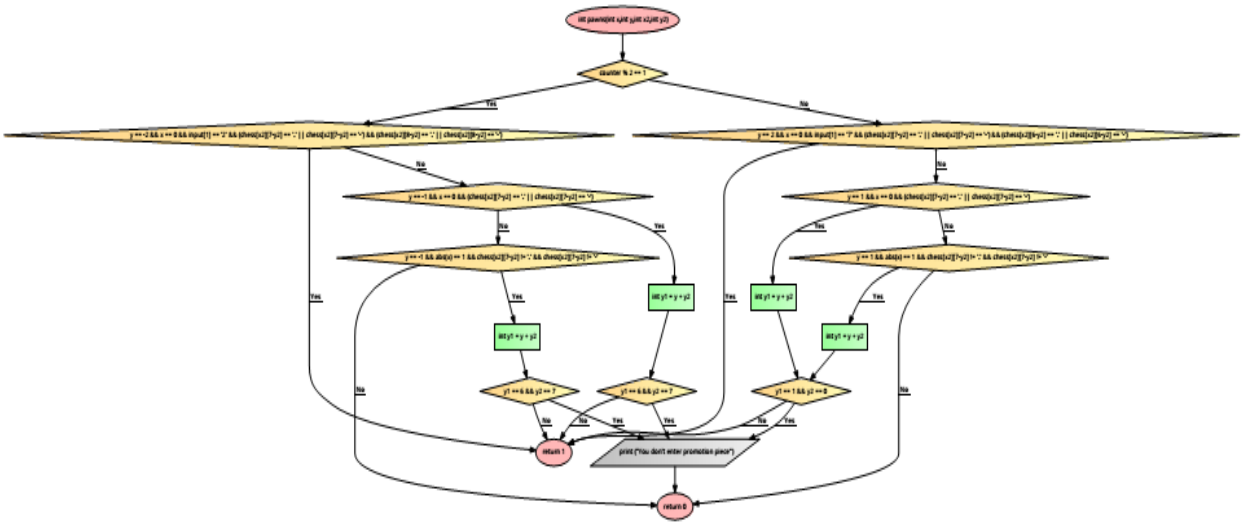
Knights:

check if the absolute difference between the beginning square and the destination square is two squares in X and one in Y or the inverse.



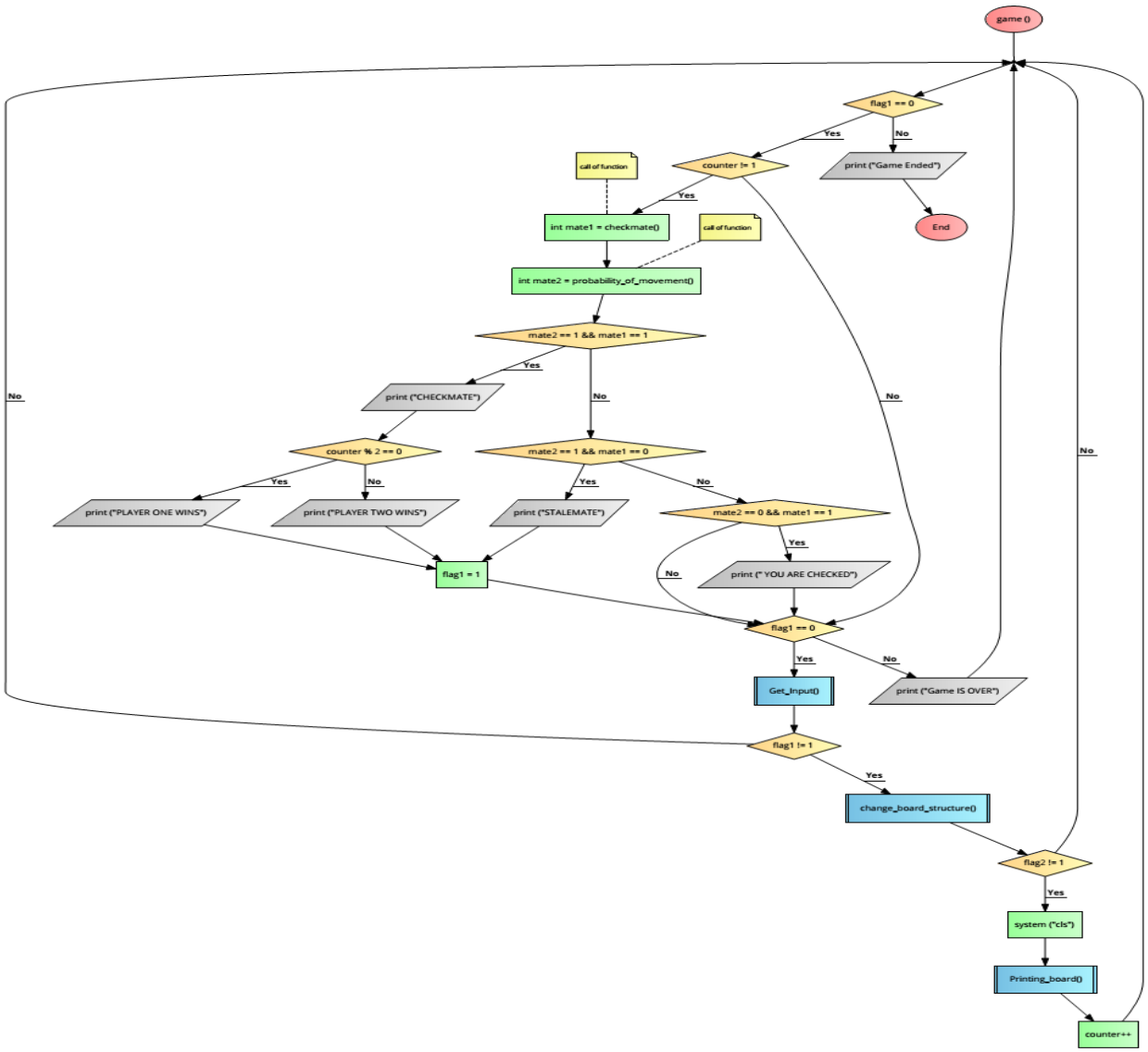
Pawns:

check if the difference between the beginning square and the destination square is one square (or two squares in its first move if the in between square is empty) in forward direction in Y and zero in X and that the destination square is empty or the difference is one square in forward direction in Y and one square in any direction in X.



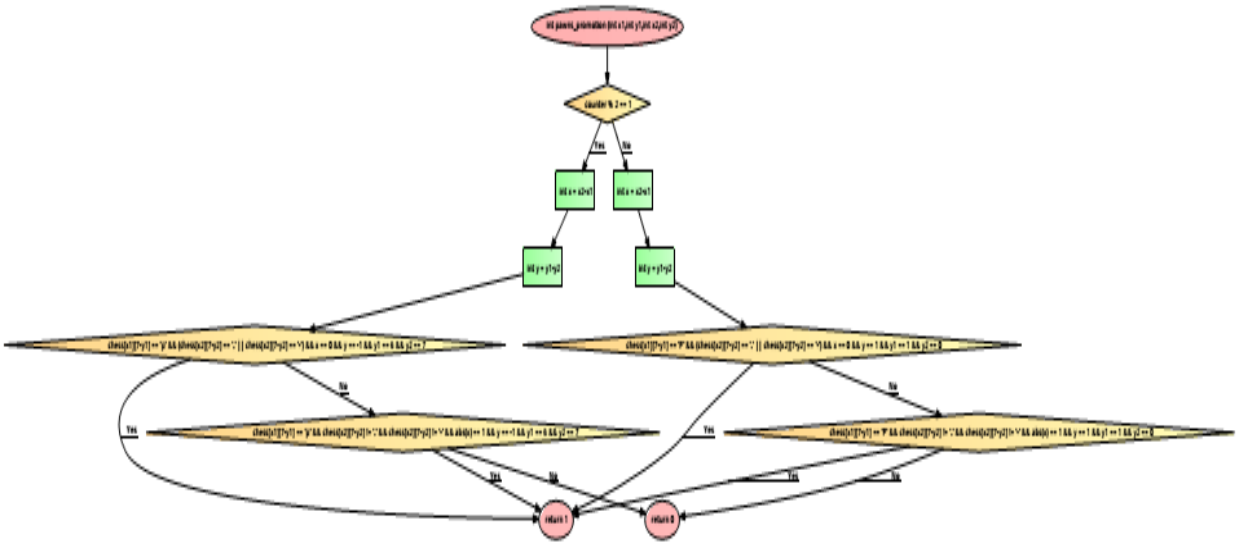
Game round:

Get Input: Takes the input from the player and check about its syntax then check if the player is moving his own pieces and that the destination square isn't containing one of its pieces then go to the validation of movement functions.
Movement functions: check if the move of this piece is available or not and then check if there is something on their way (e.g. Bishops (), Rooks () …), then check after this motion if your king is checked or not.
After each input, we check about all the possible moves for the enemy and if there is no possible move and he is checked then it is checkmate, otherwise it is stalemate and the game is ended.



Promotion:

it doesn't allow promotion unless the pawn has only one step to reach the other end of the board.



Possible moves:

it search for all your pieces then tries to move each one of them in the "64 squares in the board", if the motion is valid then it checks after doing it if the king is checked or not.