Function checked : with parameters (p = player):

Store the pieces in the structure

If p = 'b' :

current = white pieces places

(i,j) = black king place

o = 'w'

else :

current = black pieces places

(i,j) = white king place

o = 'b'

number of pieces making check = 0

r = 0

for (a,b) in current :

if piece in (a,b) can go to (i,j) :

number of pieces making check +1

place of check by piece = (a,b)

r=1

Return r

Function tempMoveCheck : with parameters(p=player , (movej,movei)=start , (movefj,movefi)=end , pro=promotion piece (which is = null if not promoted)):

If movej, movei, movefj and moveri not between 0 and 7 :

Return 1

Store check by piece

if piece in (movej,movei) can go to (movefj,movefi) :

move piece in (movej,movei) to (movefj,movefi)

r = checked or not

restore the data that was before the move using undoRedo function with parameter 'c'

restore the check by piece

Return r

Else : return 1

Function checkmate : with parameters (p = player) :

If p = 'b' :

current = black pieces places

(i,j) = black king place

o = 'w'

else :

current = white pieces places

(i,j) = white king place

o = 'b'

loop on the 9 places around the king:

if king can move to the place:

Return 0

if number of pieces making check > 1 :

Return 1

Else : (one piece making check)

(ich,jch) = place of check by piece

For (a,b) in current :

If piece in(a,b) can go to (ich,jch) : (can eat the check by piece)

Return 0

If check by piece = knight :

Return 1 (can't interrupt the path)

For place (i,j) in the path between the king and the check by piece :

For (a,b) in current :

If piece in (a,b) can go to (i,j) :

Return 0 (path can be interrupted)

Return 1

Function stalemate : with parameters (p = player):

If there are lack of checkmate material for white and black :

Return 1

If p = 'b' :

current = black pieces places

(i,j) = black king place

else :

current = white pieces places

(i,j) = white king place

loop on the 9 places around the king:

if king can move to the place:

Return 0

For (i,j) in current :

If piece in (i,j) can move :

Return 0

Return 1

Function storemove : with parameters (p=player , ifchecked , startorPlay):

Create the linked list node t

Store the board and data in t

If startorPlay = 's' : (start new game)

Previous, next of t = NULL

head = t

current = t

else if startorPlay = 'p' : (play in the game)

next of current = t

previous of t = current

next of t = NULL

current = t

Function undoRedo : with parameters (unRedo , pointer p =pointer on player , pointer ifcheck):

If unRedo = 'u' : (making undo)

If previous of current not NULL :

current = previous of current

Else : return 0 (can't do undo)

Else if unRedo = 'r' : (making redo)

If next of current not NULL :

current = next of current

Else : return 0 (can't do redo)

Else if unredo = 'c' : stay at the current

Board and data = data in current node

Return 1

Main or the game loop:

Exit = 0

X= 0 (refer to if move will happen)

piece = white

white died counter, black died counter = 0

while true :

scan sl

if sl = "load":

call load() function

break

else if sl = "start" :

break

else: print "Enter correctly"

store the start with storemove() function

while not exit :

call CheckCastling() function

print the piece

while true :

scan the move

(movej,movei) = start position

(movefj,movefi) = start position

if move[0] and move[2] between a,h and move[1] and move[3] between 1,8:

if not tempMoveCheck() on piece from(movej,movei) to(movefj,movefi):

if Checkmovement() from(movej,movei) to(movefj,movefi):

x=1

break

else if move= "save":

call save() function

print "continue or not"

scan t

if t= "y" :

exit, x = 0

call printBoard() function

break

else if t= "n" :

exit =1

x=0

break

else if move = "undo" :

x=0

if undoRedo() with parameter 'u' : (undo done)

call printBoard() function

if checked :

print "checked"

else : print "can't do undo"

break

else if move = "redo" :

x=0

if undoRedo() with parameter 'r' : (redo done)

call printBoard() function

if checked :

print "checked"

else : print "can't do redo"

break

if x=1 : (move will happen)

call movement function from(movej,movei) to(movefj,movefi)

call printBoard() function

switch players

call storemove() functions with parameter 'p' (play in game)

if player is checked :

if there a checkmate :

print "Chackmate"

print : player wins

exit = 1

else :

print "Check"

else if there stalemate :

print : "Draw"

exit = 1