**ABS Means absolute value (Positive no matter what)**

**BEGIN** MAINPROGRAM()

**SET** board to LIST [[1, 2, 3, 4, 5, 6, 7, 8] \* 8 ]

**END** MAINPROGRAM

**BEGIN** resetboard()

**FOR** ranks in range 8 **DO**

**FOR** files in range 8 **DO**

**SET** board(rank(file)) to NULL

**ENDFOR**

**ENDFOR**

**FOR** files in range 8 **DO**

**SET** board(2(files) to “WP”

**SET** board(7(files) to “BP”

**ENDFOR**

**SET** board(1(1)) to “WR”

**SET** board(1(2)) to “WN”

**SET** board(1(3)) to “WB”

**SET** board(1(4)) to “WQ”

**SET** board(1(5)) to “WK”

**SET** board(1(6)) to “WB”

**SET** board(1(7)) to “WN”

**SET** board(1(8)) to “WR”

**SET** board(8(1)) to “BR”

**SET** board(8(2)) to “BN”

**SET** board(8(3)) to “BB”

**SET** board(8(4)) to “BQ”

**SET** board(8(5) to “BK”

**SET** board(8(6)) to “BB”

**SET** board(8(7)) to “BN”

**SET** board(8(8)) to “BR”

**END** resetboard()

**BEGIN** convertindextonotation(rank, file)

“a”: 1

“b”: 2

“c” : 3

“d” : 4

“e” : 5

“f” : 6

“g” : 7

“h” : 8

**END** convertindextonotation(rank, file)

**BEGIN** checkmovevalidity[ piececode, piecelocation, newlocation]

**SET** validity to FALSE

**CASE WHERE** piececode is

“rook” : **SET** validity to checkmoveRook[ piecelocation, newlocation]  
 **BREAK**

“bishop” : **SET** validity to checkmoveBishop[ piecelocation, newlocation]

**BREAK**

“Knight” : **SET** validity to checkmoveKnight[ piecelocation, newlocation]

**BREAK**

“pawn”: **SET** validity to checkmovePawn[ piecelocation, newlocation]

**BREAK**

“king”: **SET** validity to checkmoveKing[ piecelocation, newlocation]

**BREAK**

“queen”: **SET** validity to checkmoveQueen[ piecelocation, newlocation]

**BREAK**

**RETURN** validity

**END** checkmovevalidity[]

**BEGIN** checkmoveRook[startpos, endpos]

**SET** legalmove to FALSE

**IF** startpos NOT EQUAL TO endpos **THEN**

**IF** startpos(x) EQUALS endpos(x) **AND** startpos(y) NOT EQUAL endpos(y) **THEN**

**IF** checksquarecontent[endpos] THEN

**SET** legalmove to TRUE

**ENDIF**

**ELSEIF** startpos(y) EQUALS endpos(y) **AND** startpos(x) NOT EQUAL endpos(x) **THEN**

**IF** checksquarecontent[endpos] = ‘enemy’ **OR** ‘empty’ **THEN**

**SET** legalmove to TRUE

**ENDIF**

**ENDIF**

**ENDIF**

**RETURN** legalmove

**END** checkmoveRook[startpos, endpos]

**BEGIN** checkmovePawn[startpos, endpos]

**SET** Legalmove to FALSE

**IF** startpos(x) NOT EQUAL TO endpos(x) **THEN**

**IF** startpos(y) NOT EQUAL TO endpos(y) **THEN**

**IF** startpos(y) is “2” **THEN**

**IF** endpos(y) - startpos(y) = ‘2’ **OR** endpos(y) - startpos(y) = ‘1’ **THEN**

**IF** checksquarecontent[endpos] EQUALS “empty” **THEN**

**SET** legalmove to TRUE

**ENDIF**

**ENDIF**

**ENDIF**

**IF** endpos(y) – startpos(y) = ‘1’ **THEN**

**IF** checksquarecontent[endpos] EQUALS “empty” **THEN**

**SET** legalmove to TRUE

**ENDIF**

**ENDIF**

**IF** endpos(x) EQUALS startpos(x) + 1 **AND** endpos(y) EQUALS startpos(y) +1 **OR** endpos(x) EQUALS startpos(x) – 1 **AND** endpos(y) EQUALS startpos(y) + 1 **THEN**

**IF** checksquarecontent(endpos(x, y)) EQUALS “enemy” **THEN**

**SET** Legalmove to TRUE

**ENDIF**

**ENDIF**

**ENDIF**

**ENDIF**

**RETURN** legalmove

**END** checkmovePawn

**BEGIN** checkmoveBishop[startpos, endpos]

**SET** legalmove to FALSE

**IF** checksquarecontent(endpos) EQUALS “enemy” **OR** “empty” **THEN**

**IF** startpos(x,y) IS NOT EQUAL TO endpos(x,y) **THEN**

**IF** ABS(startpos(x)) – ABS(endpos(x)) EQUALS ABS(startpos(y)) – ABS(endpos(y)) **THEN**

**SET** Legalmove to TRUE

**ENDIF**

**ENDIF**

**ENDIF**

**RETURN** legalmove

**END** checkmoveBishop[startpos, endpos]

**BEGIN** checkmoveQueen[startpos, endpos]

**SET** legalmove to FALSE

**IF** checkmoveRook[startpos, endpos] **THEN**

**SET** Legalmove to TRUE

**ENDIF**

**IF** checkmoveBishop[startpos, endpos] **THEN**

**SET** legalmove to TRUE

**ENDIF**

**RETURN** legalmove

**END** checkmoveQueen