*Software Design Document*

*- Chinese Chess*

CS3 衡雨昭 Stefan Heng

Mon. Dec. 12th 2016

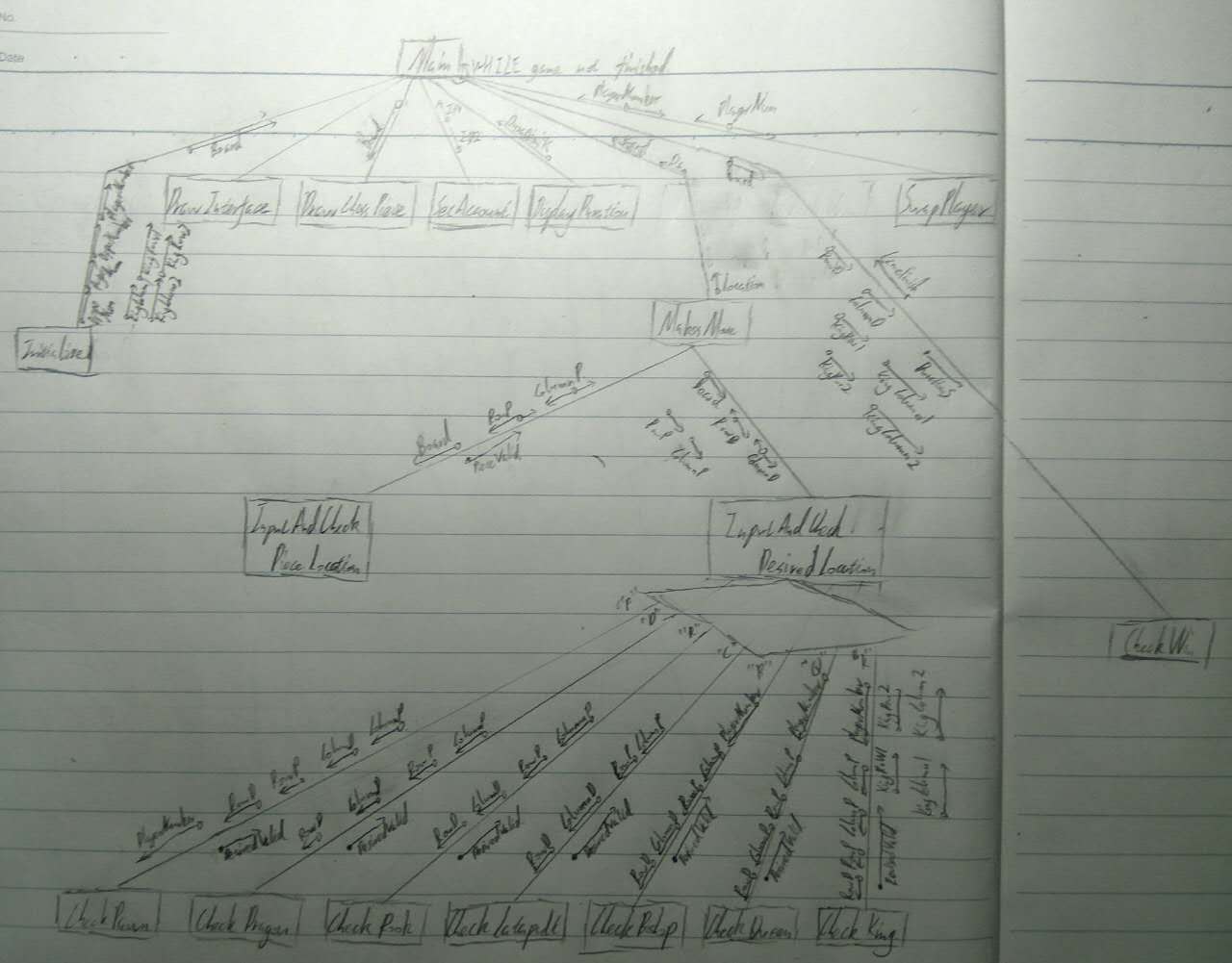
# Introduction/Overview

This program will allow 2 players to play Chinese chess several times as long as they want. Players will be able to see the duration of each game, allow several account names with their records including who wins, each move, and times of winning in an array format.

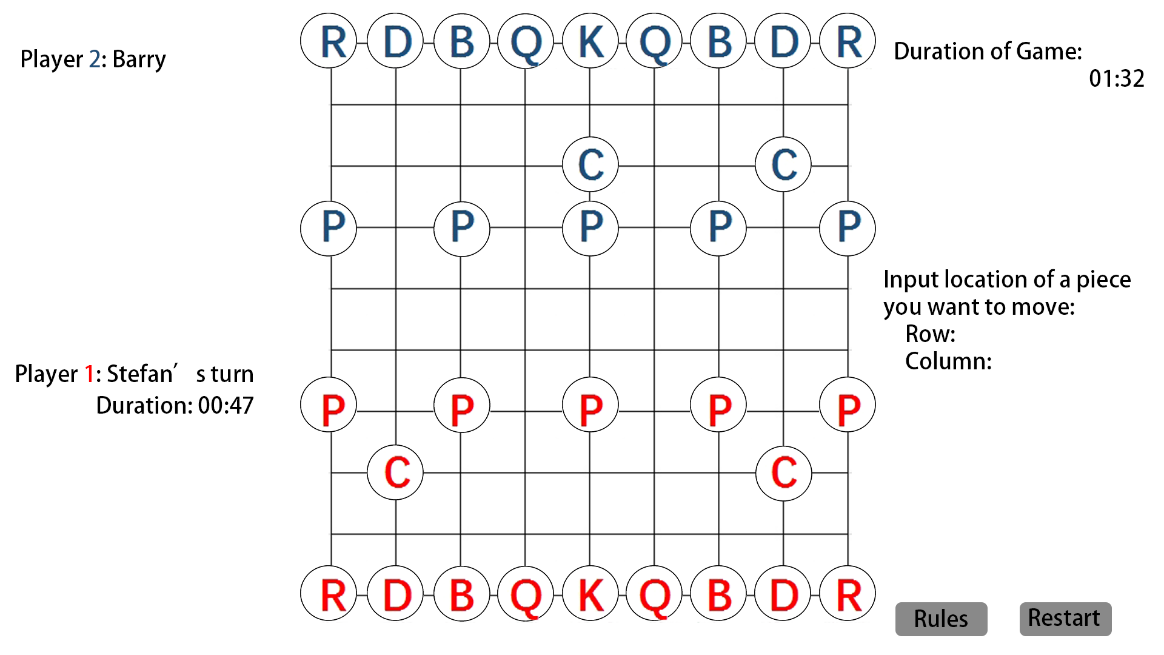
The coding will be in python, including the interface using the ‘turtle’ function. The program will be divided into procedures including loginAccount, Main, DurationOutput and Database.

# Structure Chart & Interface

* Structure Chart



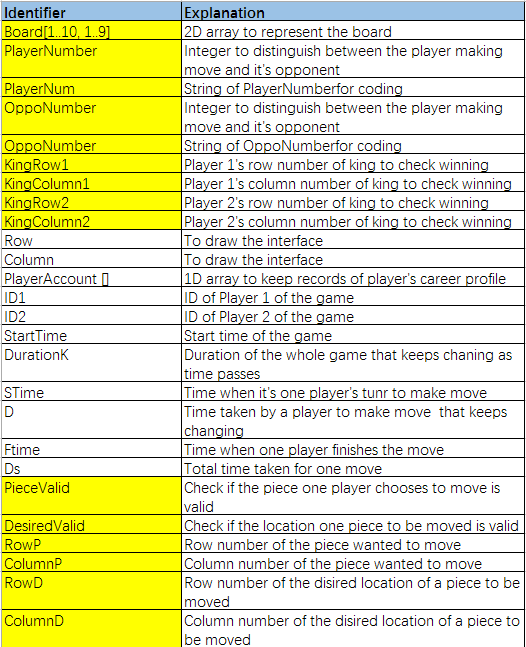
* Interface

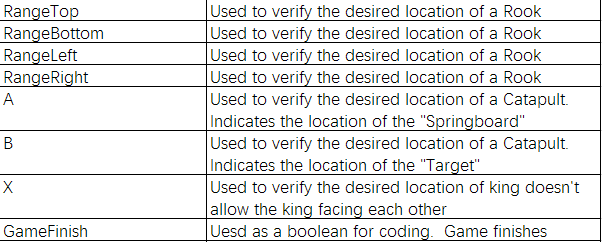


Above is the most frequently used interface, where the chess playing takes place.

# Identifier Tables

(Globals are shown by yellow background)





# Pseudocode & Flowchart

* PROCEDURE Intro

(IF BUTTON “Rules” pressed whenever) DISPLAY “

Welcome to Chinese Chess game!

K stands for King which can move 1 unoccupied square horizontally or vertically within the 9 squares; the 2 kings can’t face each other

Q stands for Queen which can move 1 unoccupied square diagonally within the 9 squares;

B stands for Bishop which can move 2 unoccupied squares diagonally within one’s side;

D stands for dragon which can move diagonally in a 2\*3 squares but can’t move so if there’s one chess piece next to it in the direction of “3”;

R stands for rook which can move any number of unoccupied squares horizontally or vertically but can’t go across a piece;

C stands for catapult which can move any number of unoccupied squares horizontally or vertically as long as there’s one chess piece between the move;

P stands for pawn which can move forwards in one’s own side and can move both forwards and horizontally on the opposite sides.

The chess piece of one side is eaten when chess piece from the other side moves to its position.”

Player on the red side starts first. Each player move their chess piece once.”

Player’s goal is to eat the opponent’s king, which decides the winner.

Each move is done by first inputting the row and column number of the location of a piece you want to move, then inputting desired location.”

ENDPROCEDURE

* PROCEDURE Restart

(IF BUTTON “Rules” pressed whenever) Restart the program from InitializeBoard

* PROCEDURE Initialize

Board = [

[R2, D2, B2, Q2, K2, Q2, B2, D2, R2],

[\_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_],

[\_\_, C2, \_\_, \_\_, \_\_, \_\_, \_\_, C2, \_\_],

[P2, \_\_, P2, \_\_, P2, \_\_, P2, \_\_, P2],

[\_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_],

[\_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_],

[P1, \_\_, P1, \_\_, P1, \_\_, P1, \_\_, P1],

[\_\_, C1, \_\_, \_\_, \_\_, \_\_, C1, \_\_, \_\_],

[\_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_, \_\_],

[R1, D1, B1, Q1, K1, Q1, B2, D2, R2]]

PlayerNumber ← 1

PlayerNum ← “1”

OppoNumber ← 2

OppoNum ← “2”

KingRow1 = 1

KingColumn1 = 5

KingRow2 = 10

KingColumn2 = 5

ENDPROCEDURE

* PROCEDURE DrawInterface

DRAW 9\*8 gird

DISPLAY 1...10, 1...9 at intersections

DRAW BUTTON “Rules”, “Start”, “Restart”

DISPLAY “Player 1:”, “Player 2:”

ENDPROCEDURE

* PROCEDURE DrawChessPiece

FOR Row ← 1 TO 10

FOR Column 1 TO 9

WHILE Board[Row][Column] == “\_\_”:

IF “1” in Board[Row][Column]

THEN

Draw circle on intersection [Row][Column]

Display Board[Row][Column] in red on intersection [Row][Column]

ELSE

Draw circle on intersection [Row][Column]

Display Board[Row][Column] in blue on intersection [Row][Column]

ENDIF

ENDWHILE

ENDFOR

ENDFOR

ENDPROCEDURE

* PROCEDURE SetAccount

PlayerAccount = []

DISPLAY “Player 1 set account: “

DRAW BUTTON “Register”, “Login”

IF “Register” pressed

THEN

INPUT ID1

STORE ID1 in PlayerAccount

ENDIF

IF “Login” pressed

THEN

INPUT ID1

WHILE ID1 in PlayerAccount = FALSE

INPUT ID1

STORE ID1 in PlayerAccount

ENDWHILE

ENDIF

DRAW BUTTON “Register”, “Login”

IF “Register” pressed

THEN

INPUT ID2

STORE ID2 in PlayerAccount

ENDIF

IF “Login” pressed

THEN

INPUT ID2

WHILE ID1 in PlayerAccount = FALSE

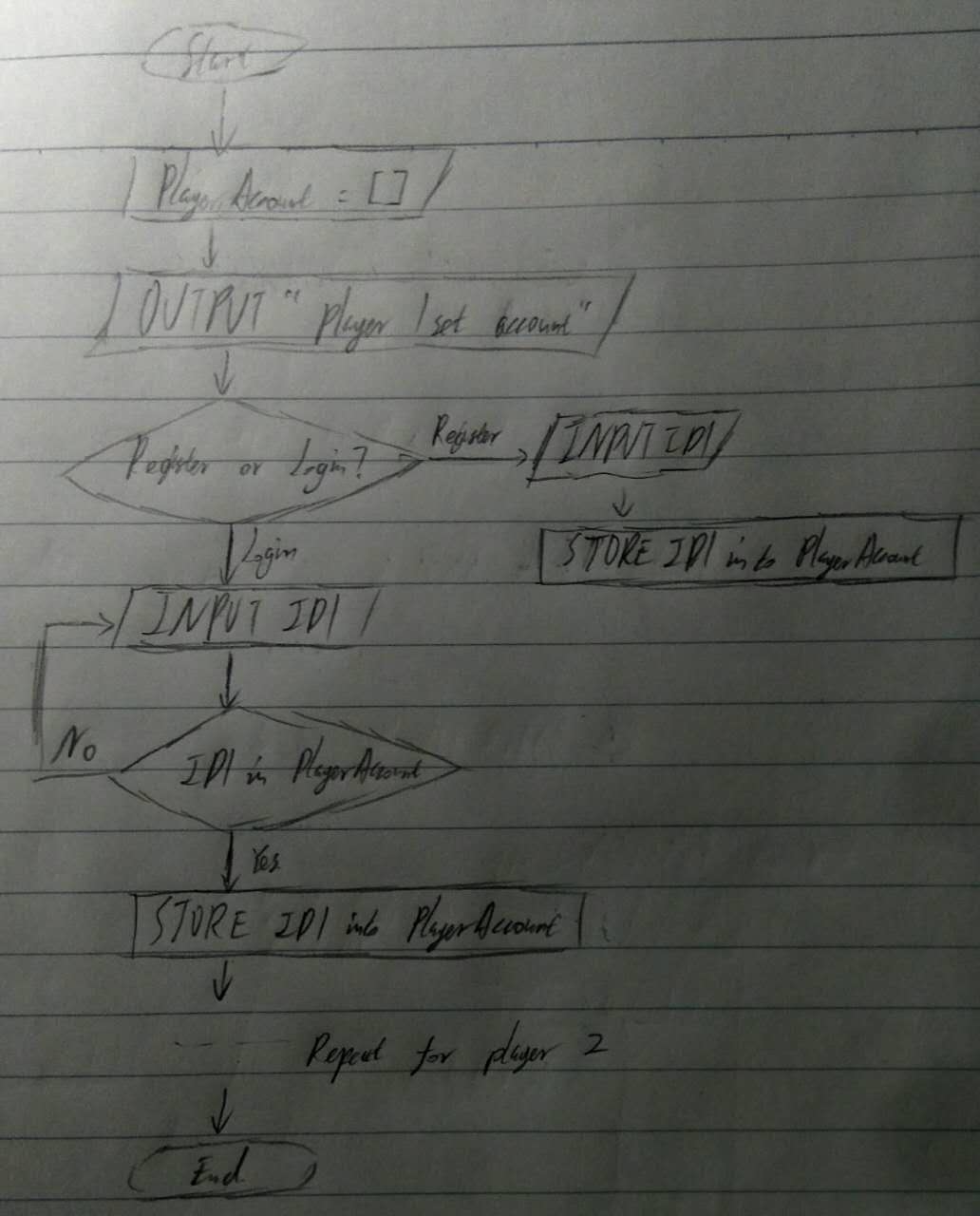
INPUT ID2

STORE ID2 in PlayerAccount

ENDWHILE

ENDIF

ENDPROCEDURE



* PROCEDURE DisplayDuration

StartTime = timenow (IF BUTTON “Start” pressed)

DurationK = timenow – StartTime

DISPLAY DurationK

ENDPROCEDURE

* PROCEDURE MakesMove

STime = timenow

D = timenow – STime

WHILE PieceValid = False

CALL InputAndCheckPieceLocation

IF BUTTON “Cancel” pressed

THEN

RESTART InputAndCheckPieceLocation

ENDIF

ENDWHILE

WHILE DesiredValid = False

CALL InputAndCheckDesiredLocation

IF BUTTON “Cancel” pressed

THEN

RESTART InputAndCheckPieceLocation

ENDIF

ENDWHILE

FTime = timenow

Ds = FTime - STime

DISPLAY Ds

STORE Ds

Board[RowD][ColumnD] = Board[RowP][ColumnP]

IF PlayerNumber = 1

THEN

Draw circle on intersection [RowD][ColumnD]

Display Board[RowD][ColumnD] in red on intersection [RowD][ColumnD]

ELSE

Draw circle on intersection [RowD][ColumnD]

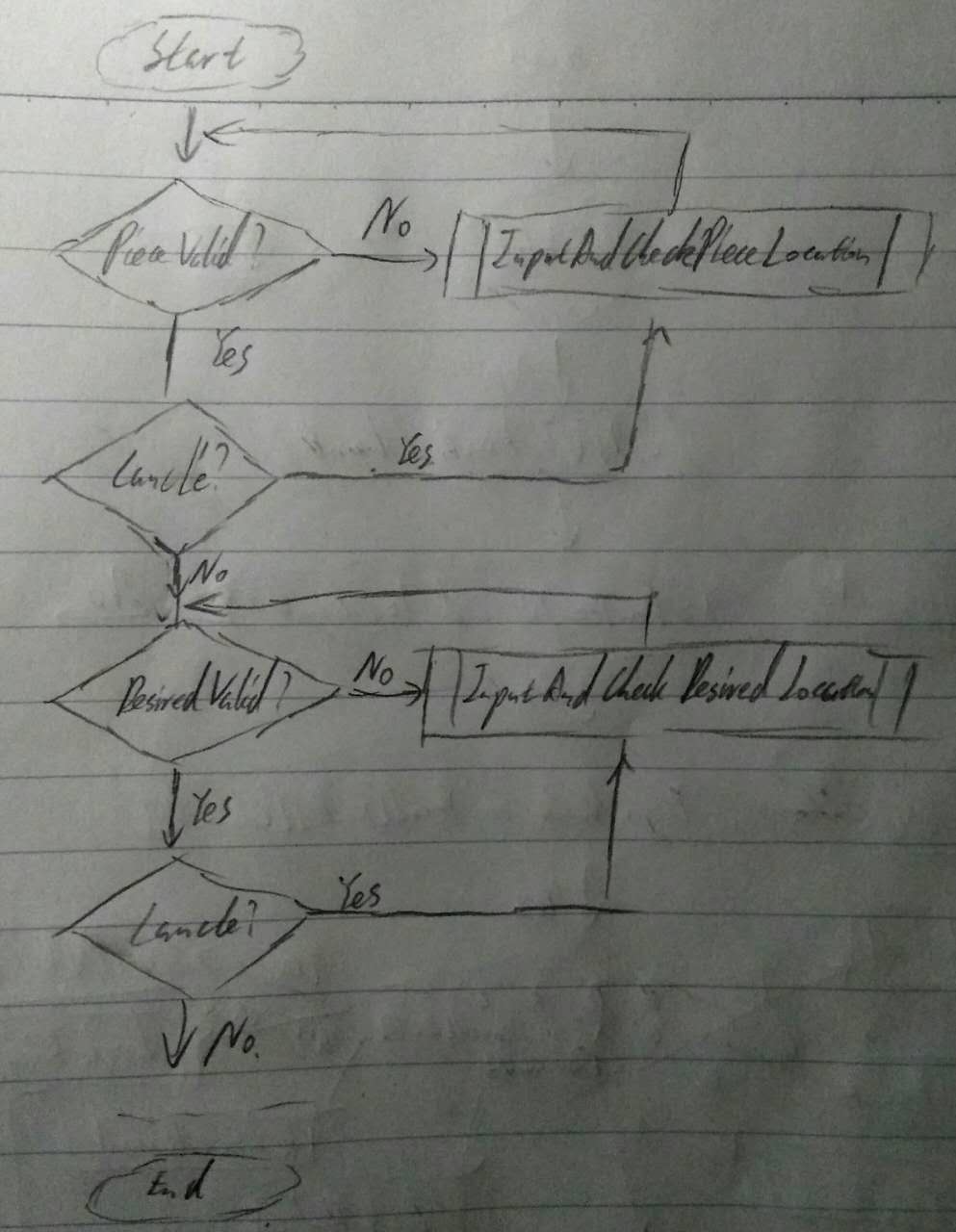
Display Board[Row][Column] in blue on intersection [RowD][ColumnD]

Board[RowP][ColumnP] = “\_\_”

ELIMINIATE circle on intersection [RowP][ColumnP]

ELIMINATE Board[RowP][ColumnP] on intersection [RowD][ColumnD]

ENDPROCEDURE



* PROCEDURE InputAndCheckPieceLocation

PieceValid ← FALSE

WHILE PieceValid = FALSE

INPUT RowP, ColumnP

IF PlayerNum in Board[RowP][ColumnP]

THEN

PieceValid ← TRUE

ELSE

OUTPUT “Invalid piece location. Please input again.”

ENDIF

ENDWHILE

STORE Location

ENDPROCEDURE

* PROCEDURE InputAndCheckDesiredLocation

DesiredValid ← FALSE

WHILE DesiredValid = FALSE

INPUT RowD, ColumnD

IF RowD >= 1 and RowD <= 10 and ColumnD >=1 and ColumnD <= 9

THEN

IF PlayerNum in Board [RowD][ColumnD] = FALSE

THEN

IF “P” in Board[RowP][ColumnP]

THEN

CALL CheckPawn

ELIF “D” in Board[RowP][ColumnP]

CALL CheckDragon

ELIF “R” in Board[RowP][ColumnP]

CALL CheckRook

ELIF “C” in Board[RowP][ColumnP]

CALL CheckCatapult

ELIF “B” in Board[RowP][ColumnP]

CALL CheckBishop

ELIF “Q” in Board[RowP][ColumnP]

CALL CheckQueen

ELIF “K” in Board[RowP][ColumnP]

CALL CheckKing

ELSE

OUTPUT “Unknown error. Please inform the staff and start over.”

ENDIF

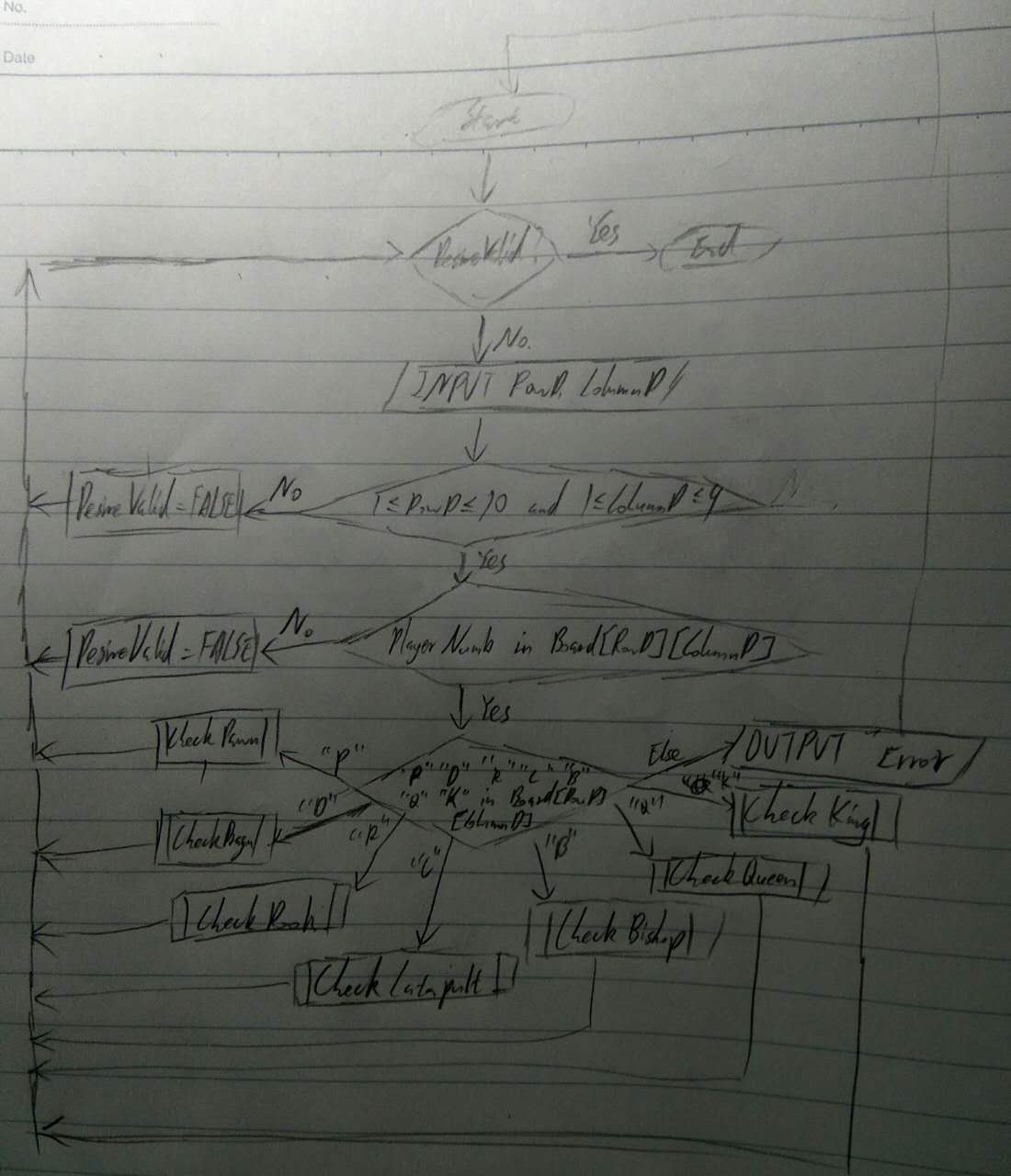
ENDIF

ENDIF

ENDWHILE

STORE Location

ENDPROCEDURE



* PROECDURE CheckPawn

IF PlayerNumber = 1

THEN

IF RowD = 5

THEN

IF RowP = 4

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELIF RowD >= 6

IF RowD = RowP + 1 and ColumnD = Column P

THEN

DesiredValid ← TRUE

ELIF RowD = RowP and ColumnD = ColumnP + 1

DesiredValid ← TRUE

ELIF RowD = RowP and ColumnD = ColumnP – 1

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELSE

IF RowD = 6

THEN

IF RowP = 7

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELIF RowD <= 5

IF RowD = RowP - 1 and ColumnD = Column P

THEN

DesiredValid ← TRUE

ELIF RowD = RowP and ColumnD = ColumnP + 1

DesiredValid ← TRUE

ELIF RowD = RowP and ColumnD = ColumnP – 1

DesiredValid ← TRUE

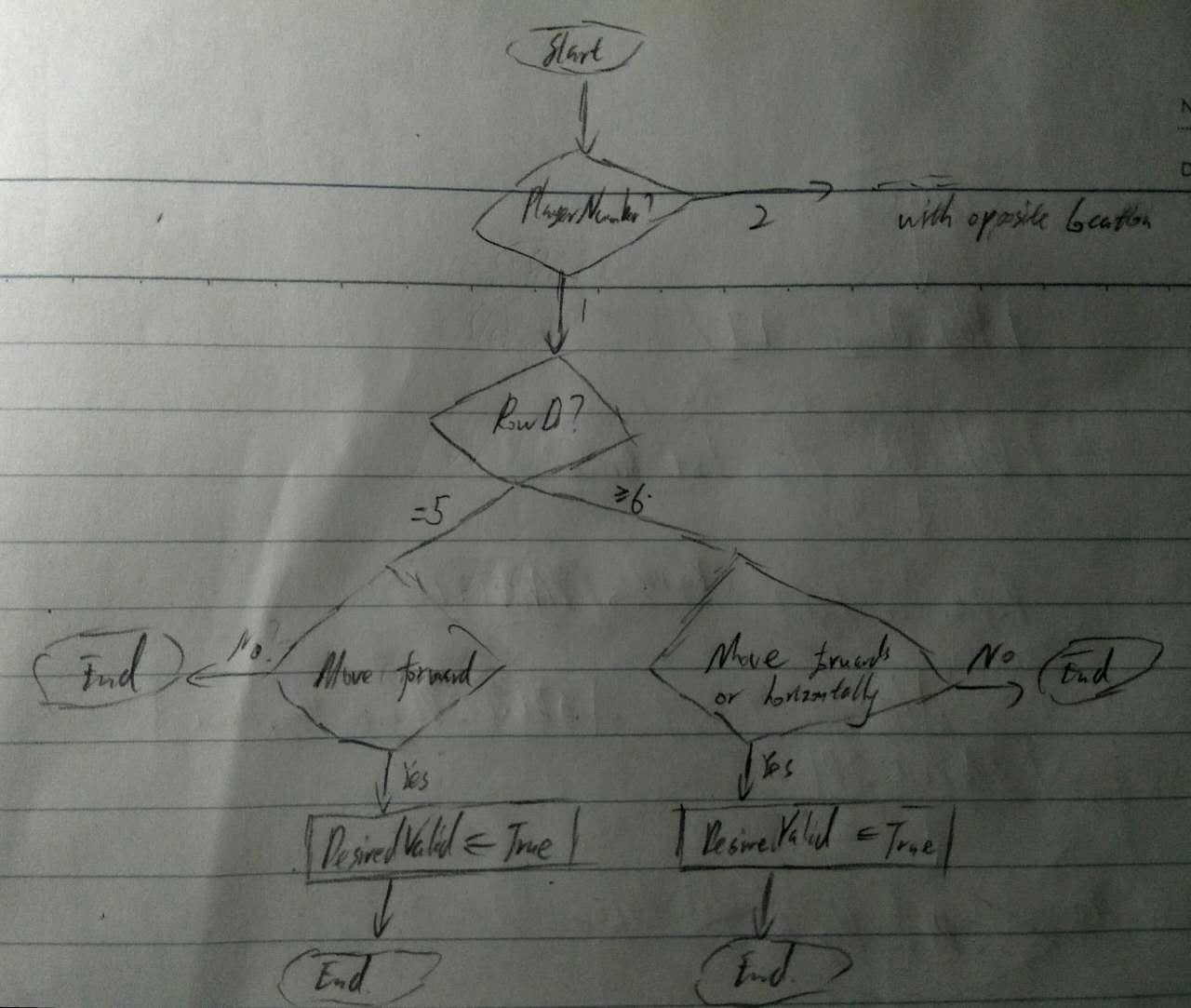
ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ENDPROCEDURE



* PROCEDURE CheckDragon

IF RowD = RowP + 1 or RowD = RowP - 1

THEN

IF ColumnD = ColumnP + 2

THEN

IF Board[RowP][ColumnP+1] = “\_\_”

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELIF ColumnD = ColumnP – 2

IF Board[RowP][ColumnP-1] = “\_\_”

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ELIF ColumnD = ColumnP + 1 or ColumnD = ColumnP – 1

IF RowD = RowP + 2

THEN

IF Board [RowP + 1] [ColumnP] = “\_\_”

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELIF RowD = RowP – 2

THEN

IF Board [RowP - 1] [ColumnP] = “\_\_”

THEN

DesiredValid ← TRUE

ELSE

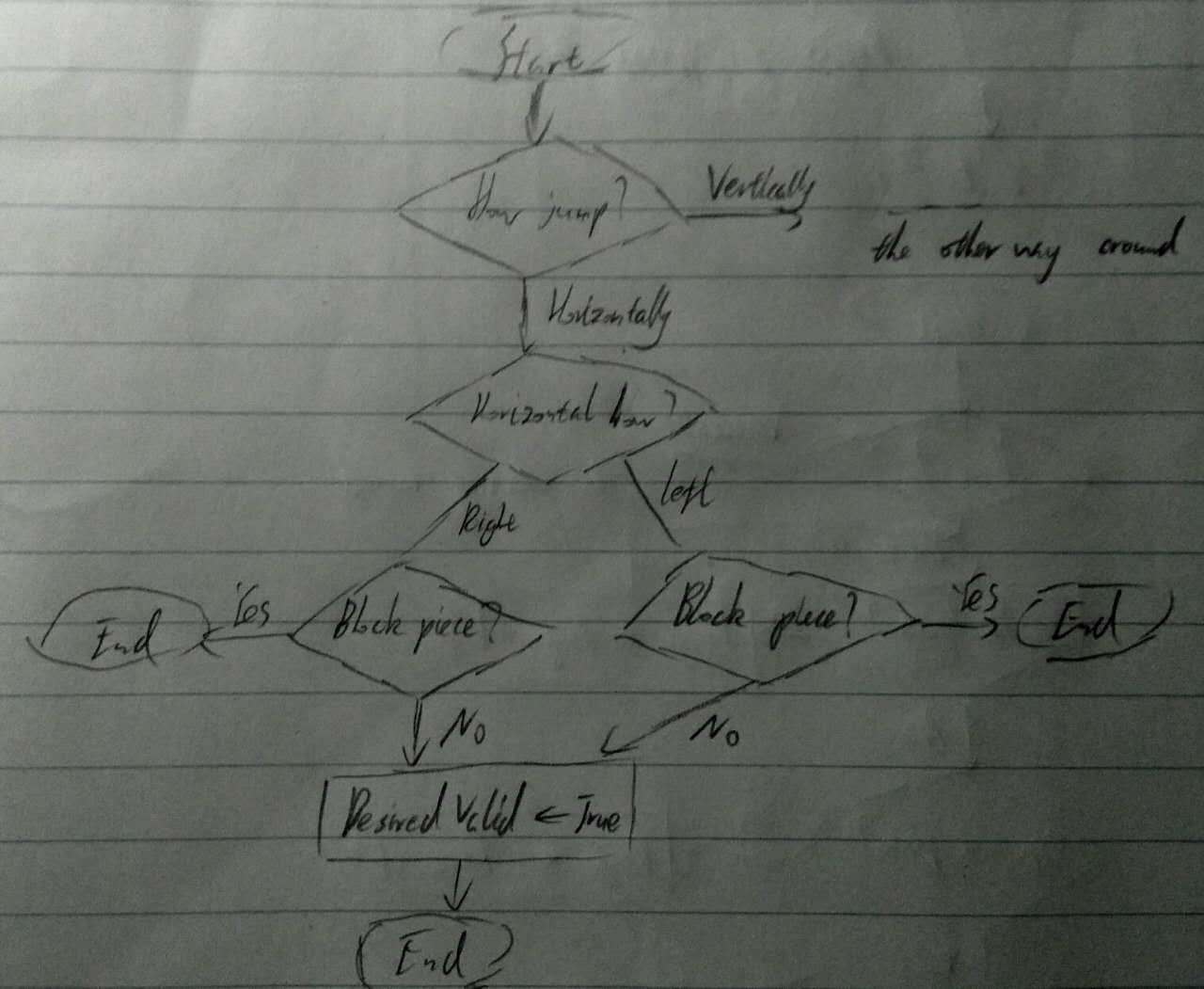
OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ENDIF

ENDPROCEDURE



* PROCEDURE CheckRook

RangeTop ← RowP

RangeBottom ← RowP

RangeLeft ← ColumnP

RangeRight ← ColumnP

IF RowD = RowP

THEN

WHILE Board [RowP] [RangeLeft - 1] = “\_\_”

RangeLeft ← RangeLeft - 1

ENDWHILE

IF OppoNum in Board [RowP] [RangeLeft - 1]

THEN

RangeLeft ← RangeLeft - 1

ENDIF

WHILE Board [RowP] [RangeRight + 1] = “\_\_”

RangeRight ← RangeRight + 1

ENDWHILE

IF OppoNum in Board [RowP] [RangeRight + 1]

THEN

RangeRight ← RangeRight + 1

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

IF ColumnD >= RangeLeft and ColumnD <= RangeRight

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELIF ColumnD = ColumnP

WHILE Board [RangeTop + 1] [ColumnP] = “\_\_”

RangeTop ← RangeTop + 1

ENDWHILE

IF OppoNum in Board [RangeTop + 1] [Column]

THEN

RangeTop ← RangeTop + 1

ENDIF

WHILE Board [RangeBottom - 1] [ColumnP] = “\_\_”

RangeBottom ← RangeBottom - 1

ENDWHILE

IF OppoNum in Board [RangeBottom - 1] [ColumnP]

THEN

RangeBottom ← RangeBottom - 1

ENDIF

IF RowD >= RangeBottom and RowD <= RangeTop

THEN

DesiredValid ← TRUE

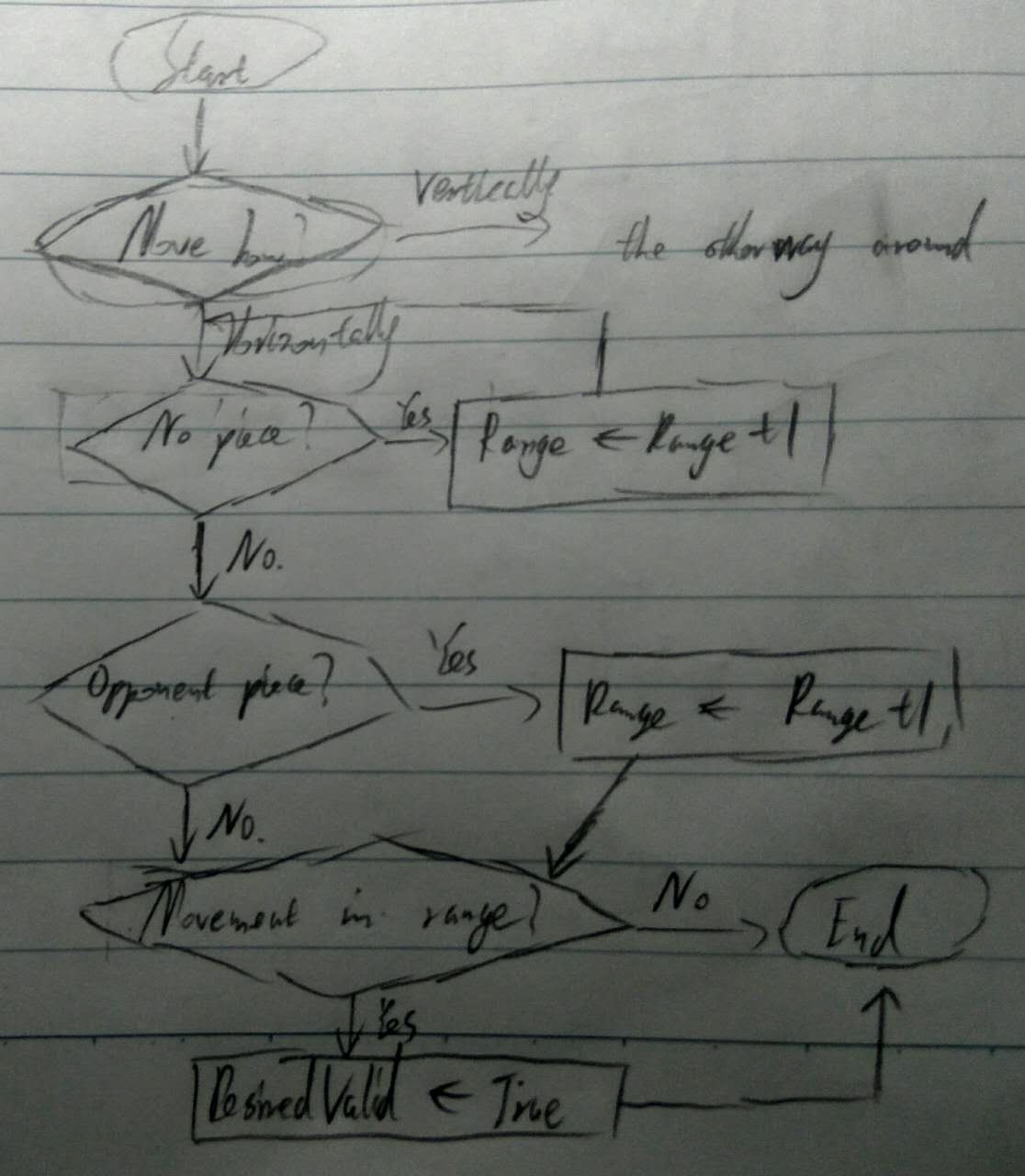
ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ENDPROCEDURE



* PROCEDURE CheckCatapult

IF ColumnD = ColumnP

THEN

IF RowD > RowP

THEN

A ← RowP

WHILE Board[A][ColumnD] = “\_\_”

A ← A + 1

ENDWHILE

IF “1” in Board [A + 1] [ColumnD] or “2” in Board [A + 1] [ColumnD]

THEN

A ← A + 1

B ← A + 1

WHILE Board[B][ColumnD] = “\_\_”

B ← B + 1

ENDWHILE

B ← B + 1

IF OppoNumber in Board [B] [ColumnD] and B = RowD

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ELIF RowD < RowP

A ← RowP

WHILE Board[A][ColumnD] = “\_\_”

A ← A - 1

ENDWHILE

IF “1” in Board [A - 1] [ColumnD] or “2” in Board [A - 1] [ColumnD]

THEN

A ← A - 1

B = A - 1

WHILE Board[B][ColumnD] = “\_\_”

B ← B - 1

ENDWHILE

B ← B – 1

IF OppoNumber in Board [B] [ColumnD] and B = RowD

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ENDIF

ENDIF

IF RowD = RowP

THEN

IF ColumnD > ColumnP

THEN

A ← ColumnP

WHILE Board[RowD][A] = “\_\_”

A ← A + 1

ENDWHILE

IF “1” in Board [RowD] [A + 1] or “2” in Board [RowD] [A + 1]

THEN

A ← A + 1

B ← A + 1

WHILE Board[RowD] [ B] = “\_\_”

B ← B + 1

ENDWHILE

B ← B + 1

IF OppoNumber in Board [RowD] [B] and B = ColumnD

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ELIF ColumnD < ColumnP

A ← ColumnP

WHILE Board[RowD][A] = “\_\_”

A ← A - 1

ENDWHILE

IF “1” in Board [RowD] [A - 1] or “2” in Board [RowD] [A - 1]

THEN

A ← A - 1

B ← A - 1

WHILE Board[RowD] [ B] = “\_\_”

B ← B + 1

ENDWHILE

B ← B + 1

IF OppoNumber in Board [RowD] [B] and B = ColumnD

THEN

DesiredValid ← TRUE

ELSE

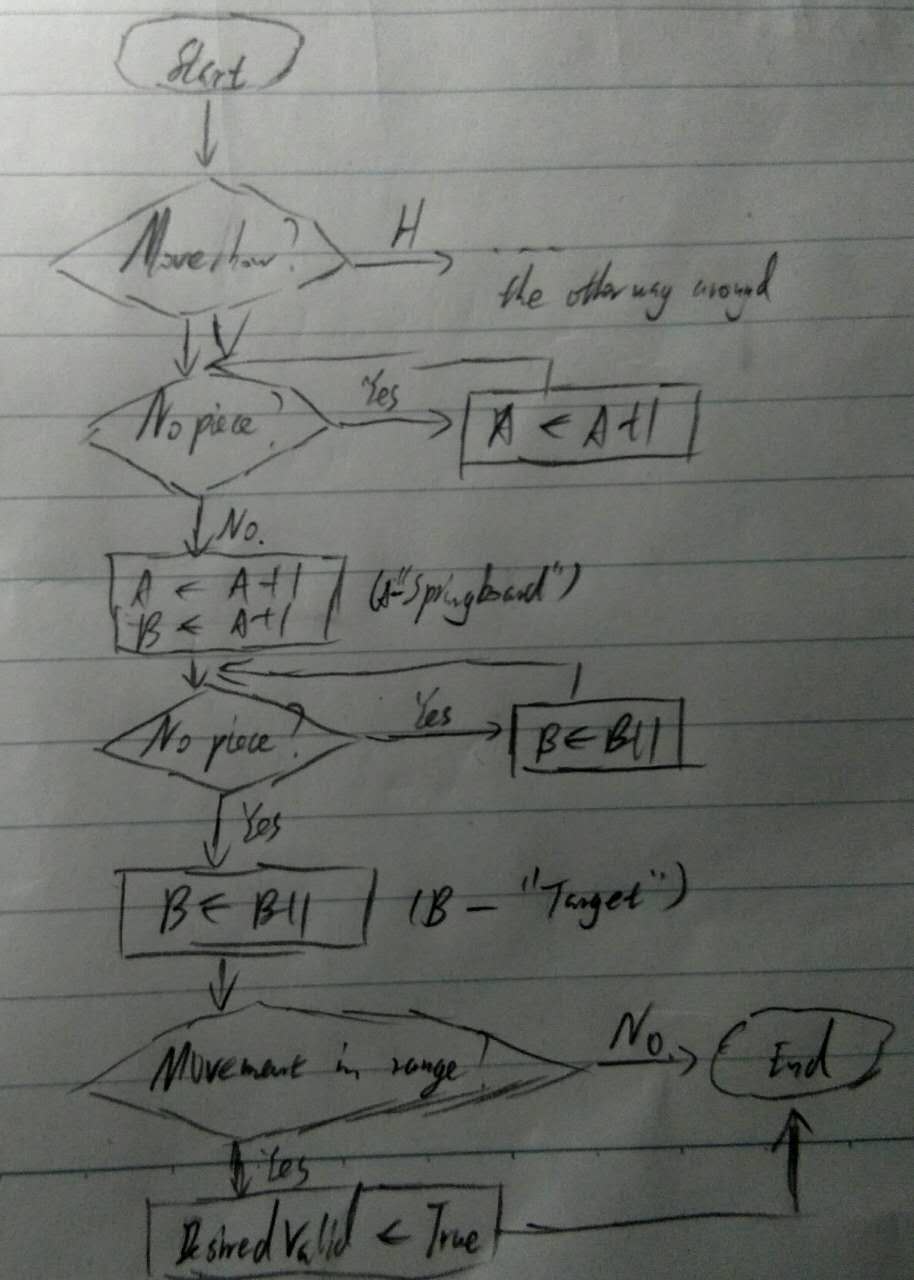
OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ENDIF

ENDPROCEDURE



* PROCEDURE CheckBishop

IF PlayerNumber = 1

THEN

IF RowD >= 6 and RowD <= 10 and ColumnD <= 1 and ColumnD >= 9

THEN

IF |RowD – RowP| = 2 and |ColumnD – ColumnP| = 2

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELSE

IF RowD >= 1 and RowD <= 5 and ColumnD <= 1 and ColumnD >= 9

THEN

IF |RowD – RowP| = 2 and |ColumnD – ColumnP| = 2

THEN

DesiredValid ← TRUE

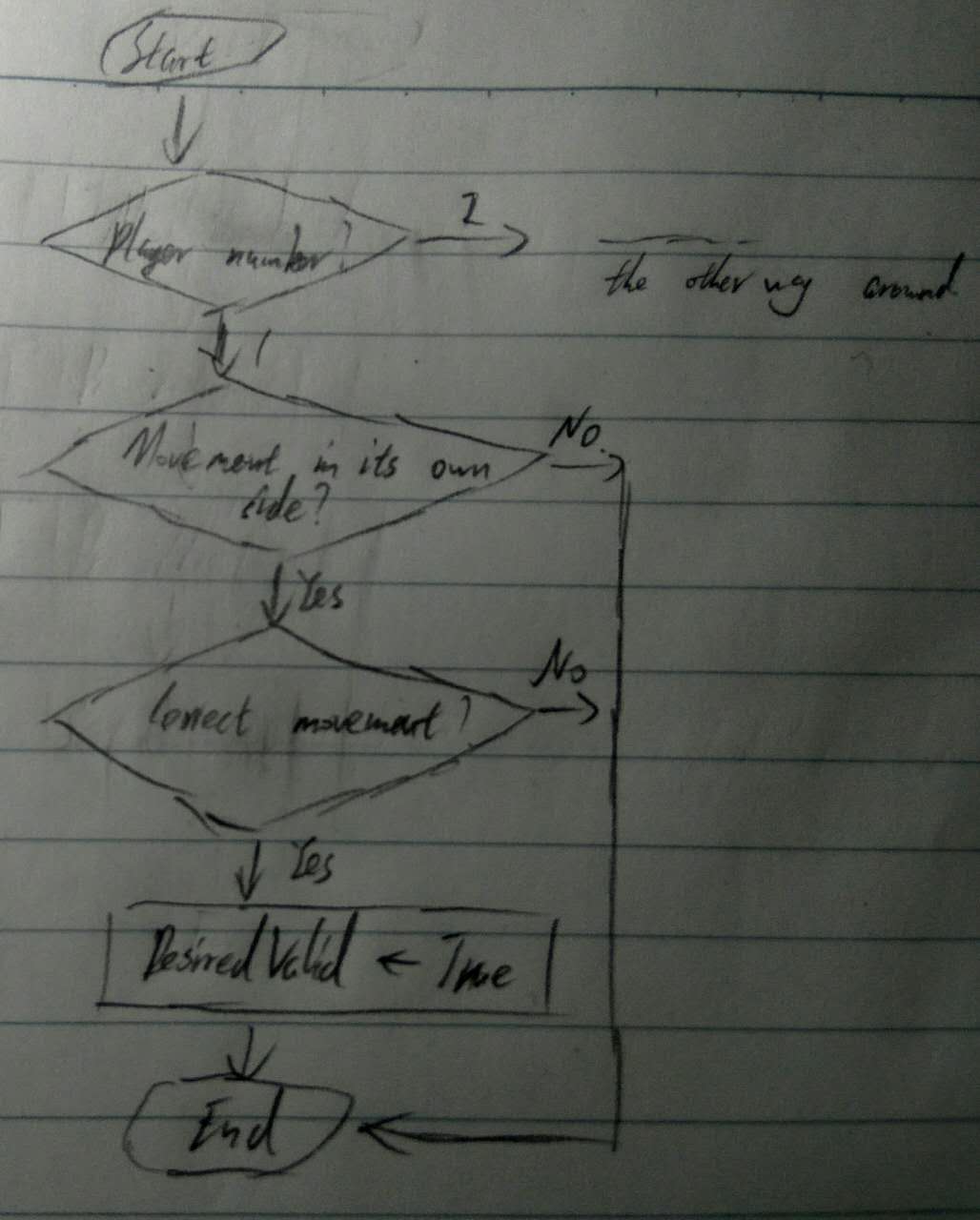
ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ENDPROCEDURE



* PROCEDURE CheckQueen

IF PlayerNumber = 1

THEN

IF RowD >= 8 and RowD <= 10 and ColumnD <= 4 and ColumnD >= 6

THEN

IF |RowD – RowP| = 1 and |ColumnD – ColumnP| = 1

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ELSE

IF RowD >= 1 and RowD <= 3 and ColumnD <= 4 and ColumnD >= 6

THEN

IF |RowD – RowP| = 1 and |ColumnD – ColumnP| = 1

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ENDIF

ENDIF

ENDPROCEDURE

Follows the same way of checking as that of Bishop.

* PROCEDURE CheckKing

Face = FALSE

IF PlayerNumber = 1

THEN

IF RowD >= 8 and RowD <= 10 and ColumnD <= 4 and ColumnD >= 6

THEN

IF |RowD – RowP| = 1 and ColumnD = ColumnP

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELIF |ColumnD – ColumnP| = 1 and RowD = RowP

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

IF ColumnD = KingColumn2

THEN

FOR X = KingRow2 + 1 TO RowD – 1

IF Board[X][ColumnD] = “\_\_” == FALSE：

DesiredValid ← FALSE

ENDIF

X = X + 1

ENDFOR

ENDIF

KingRow1 = RowD

KingColumn1 = ColumnD

ELSE

IF RowD >= 1 and RowD <= 3 and ColumnD <= 4 and ColumnD >= 6

THEN

IF |RowD – RowP| = 1 and ColumnD = ColumnP

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

ELIF |ColumnD – ColumnP| = 1 and RowD = RowP

THEN

DesiredValid ← TRUE

ELSE

OUTPUT “Invalid desired location. Please input again.”

ENDIF

IF ColumnD = KingColumn1

THEN

FOR X = RowD + 1 TO KingRow1

IF Board[X][ColumnD] = “\_\_” == FALSE：

DesiredValid ← FALSE

ENDIF

X = X + 1

ENDFOR

ENDIF

KingRow2 = RowD

KingColumn2 = ColumnD

ENDPROCEDURE

Follows the same way of checking as that of Bishop, except adding the part of checking if the 2 kings would face each other.

* PROCEDURE CheckWin

IF PlayerNumber = 1

THEN

IF KingRow2 = RowD and KingColumn2 = ColumnD

THEN

GameFinish = TRUE

DISPLAY “Player 1” ID1 “wins. Congratulations!”

DurationS = timenow

DISPLAY DurationS

ENDIF

ELSE

IF KingRow1 = RowD and KingColumn1 = ColumnD

THEN

GameFinish = TRUE

DISPLAY “Player 2” ID2 “wins. Congratulations!”

DurationS = timenow

DISPLAY DurationS

ENDIF

ENDIF

STORE DurationS

ENDPROCEDURE

* PROCEDURE SwapPlayer

IF PlayerNumber = 1

PlayerNumber ← 2

PlayerNum ← PlayerNumber format STRING

OppoNumber ← 1

OppoNum ← OppoNumber format STRING

ELSE

PlayerNumber ← 1

PlayerNum ← PlayerNumber format STRING

OppoNumber ← 2

OppoNum ← OppoNumber format STRING

ENDPROCEDURE

* PROCEDURE Main

CALL Initialize

CALL DrawInterface

CALL DrawChessPiece

CALL SetAccount

CALL DisplayDuration

WHILE GameFinish = FALSE

CALL MakesMove

CALL CheckWin

CALL SwapPlayer

ENDWHILE

CALL Intro (IF BUTTON “Rules” pressed)

CALL Restart (IF BUTTON “Restart” pressed)

ENDPROCEDURE

CALL Main