**PLEASE COPY AND PASTE THE FOLLOWING CODE INTO PYTHON FOR THE SECTION D COMPONENT OF TASK 2**

**----------------------------------------------------------------------------------------------------------------------------------**

# Skeleton Program code for the AQA COMP1 Summer 2017 examination

# this code should be used in conjunction with the Preliminary Material

# written by the AQA COMP1 Programmer Team

# developed in the Python 3.4 programming environment

BOARDDIMENSION = 8

def CreateBoard():

 Board = []

 for Count in range(BOARDDIMENSION + 1):

 Board.append([])

 for Count2 in range(BOARDDIMENSION + 1):

 Board[Count].append(" ")

 return Board

def DisplayWhoseTurnItIs(WhoseTurn):

 if WhoseTurn == "W":

 print("It is White's turn")

 else:

 print("It is Black's turn")

def GetTypeOfGame():

 TypeOfGame = input("Do you want to play the sample game (enter Y for Yes)? ")

 return TypeOfGame

def DisplayWinner(WhoseTurn):

 if WhoseTurn == "W":

 print("Black's Sarrum has been captured. White wins!")

 else:

 print("White's Sarrum has been captured. Black wins!")

def CheckIfGameWillBeWon(Board, FinishRank, FinishFile):

 if Board[FinishRank][FinishFile][1] == "S":

 return True

 else:

 return False

def DisplayBoard(Board):

 print()

 for RankNo in range(1, BOARDDIMENSION + 1):

 print(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

 print(RankNo, end=" ")

 for FileNo in range(1, BOARDDIMENSION + 1):

 print("|" + Board[RankNo][FileNo], end="")

 print("|")

 print(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

 print()

 print(" 1 2 3 4 5 6 7 8")

 print()

 print()

def CheckRedumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, ColourOfPiece):

 CheckRedumMoveIsLegal = False

 if ColourOfPiece == "W":

 if FinishRank == StartRank - 1:

 if FinishFile == StartFile and Board[FinishRank][FinishFile] == " ":

 CheckRedumMoveIsLegal = True

 elif abs(FinishFile - StartFile) == 1 and Board[FinishRank][FinishFile][0] == "B":

 CheckRedumMoveIsLegal = True

 elif FinishRank == StartRank + 1:

 if FinishFile == StartFile and Board[FinishRank][FinishFile] == " ":

 CheckRedumMoveIsLegal = True

 elif abs(FinishFile - StartFile) == 1 and Board[FinishRank][FinishFile][0] == "W":

 CheckRedumMoveIsLegal = True

 return CheckRedumMoveIsLegal

def CheckSarrumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile):

 CheckSarrumMoveIsLegal = False

 if abs(FinishFile - StartFile) <= 1 and abs(FinishRank - StartRank) <= 1:

 CheckSarrumMoveIsLegal = True

 return CheckSarrumMoveIsLegal

def CheckGisgigirMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile):

 GisgigirMoveIsLegal = False

 RankDifference = FinishRank - StartRank

 FileDifference = FinishFile - StartFile

 if RankDifference == 0:

 if FileDifference >= 1:

 GisgigirMoveIsLegal = True

 for Count in range(1, FileDifference):

 if Board[StartRank][StartFile + Count] != " ":

 GisgigirMoveIsLegal = False

 elif FileDifference <= -1:

 GisgigirMoveIsLegal = True

 for Count in range(-1, FileDifference, -1):

 if Board[StartRank][StartFile + Count] != " ":

 GisgigirMoveIsLegal = False

 elif FileDifference == 0:

 if RankDifference >= 1:

 GisgigirMoveIsLegal = True

 for Count in range(1, RankDifference):

 if Board[StartRank + Count][StartFile] != " ":

 GisgigirMoveIsLegal = False

 elif RankDifference <= -1:

 GisgigirMoveIsLegal = True

 for Count in range(-1, RankDifference, -1):

 if Board[StartRank + Count][StartFile] != " ":

 GisgigirMoveIsLegal = False

 return GisgigirMoveIsLegal

def CheckNabuMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile):

 CheckNabuMoveIsLegal = False

 if abs(FinishFile - StartFile) == 1 and abs(FinishRank - StartRank) == 1:

 CheckNabuMoveIsLegal = True

 return CheckNabuMoveIsLegal

def CheckMarzazPaniMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile):

 CheckMarzazPaniMoveIsLegal = False

 if (abs(FinishFile - StartFile) == 1 and abs(FinishRank - StartRank) == 0) or (abs(FinishFile - StartFile) == 0 and abs(FinishRank - StartRank) ==1):

 CheckMarzazPaniMoveIsLegal = True

 return CheckMarzazPaniMoveIsLegal

def CheckEtluMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile):

 CheckEtluMoveIsLegal = False

 if (abs(FinishFile - StartFile) == 2 and abs(FinishRank - StartRank) == 0) or (abs(FinishFile - StartFile) == 0 and abs(FinishRank - StartRank) == 2):

 CheckEtluMoveIsLegal = True

 return CheckEtluMoveIsLegal

def CheckMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn):

 MoveIsLegal = True

 if (FinishFile == StartFile) and (FinishRank == StartRank):

 MoveIsLegal = False

 else:

 PieceType = Board[StartRank][StartFile][1]

 PieceColour = Board[StartRank][StartFile][0]

 if WhoseTurn == "W":

 if PieceColour != "W":

 MoveIsLegal = False

 if Board[FinishRank][FinishFile][0] == "W":

 MoveIsLegal = False

 else:

 if PieceColour != "B":

 MoveIsLegal = False

 if Board[FinishRank][FinishFile][0] == "B":

 MoveIsLegal = False

 if MoveIsLegal == True:

 if PieceType == "R":

 MoveIsLegal = CheckRedumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, PieceColour)

 elif PieceType == "S":

 MoveIsLegal = CheckSarrumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile)

 elif PieceType == "M":

 MoveIsLegal = CheckMarzazPaniMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile)

 elif PieceType == "G":

 MoveIsLegal = CheckGisgigirMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile)

 elif PieceType == "N":

 MoveIsLegal = CheckNabuMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile)

 elif PieceType == "E":

 MoveIsLegal = CheckEtluMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile)

 return MoveIsLegal

def InitialiseBoard(Board, SampleGame):

 if SampleGame == "Y":

 for RankNo in range(1, BOARDDIMENSION + 1):

 for FileNo in range(1, BOARDDIMENSION + 1):

 Board[RankNo][FileNo] = " "

 Board[1][2] = "BG"

 Board[1][4] = "BS"

 Board[1][8] = "WG"

 Board[2][1] = "WR"

 Board[3][1] = "WS"

 Board[3][2] = "BE"

 Board[3][8] = "BE"

 Board[6][8] = "BR"

 else:

 for RankNo in range(1, BOARDDIMENSION + 1):

 for FileNo in range(1, BOARDDIMENSION + 1):

 if RankNo == 2:

 Board[RankNo][FileNo] = "BR"

 elif RankNo == 7:

 Board[RankNo][FileNo] = "WR"

 elif RankNo == 1 or RankNo == 8:

 if RankNo == 1:

 Board[RankNo][FileNo] = "B"

 if RankNo == 8:

 Board[RankNo][FileNo] = "W"

 if FileNo == 1 or FileNo == 8:

 Board[RankNo][FileNo] = Board[RankNo][FileNo] + "G"

 elif FileNo == 2 or FileNo == 7:

 Board[RankNo][FileNo] = Board[RankNo][FileNo] + "E"

 elif FileNo == 3 or FileNo == 6:

 Board[RankNo][FileNo] = Board[RankNo][FileNo] + "N"

 elif FileNo == 4:

 Board[RankNo][FileNo] = Board[RankNo][FileNo] + "M"

 elif FileNo == 5:

 Board[RankNo][FileNo] = Board[RankNo][FileNo] + "S"

 else:

 Board[RankNo][FileNo] = " "

def GetMove(StartSquare, FinishSquare):

 StartSquare = int(input("Enter coordinates of square containing piece to move (file first): "))

 FinishSquare = int(input("Enter coordinates of square to move piece to (file first): "))

 return StartSquare, FinishSquare

def MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn):

 if WhoseTurn == "W" and FinishRank == 1 and Board[StartRank][StartFile][1] == "R":

 Board[FinishRank][FinishFile] = "WM"

 Board[StartRank][StartFile] = " "

 elif WhoseTurn == "B" and FinishRank == 8 and Board[StartRank][StartFile][1] == "R":

 Board[FinishRank][FinishFile] = "BM"

 Board[StartRank][StartFile] = " "

 else:

 Board[FinishRank][FinishFile] = Board[StartRank][StartFile]

 Board[StartRank][StartFile] = " "

if \_\_name\_\_ == "\_\_main\_\_":

 Board = CreateBoard() #0th index not used

 StartSquare = 0

 FinishSquare = 0

 PlayAgain = "Y"

 while PlayAgain == "Y":

 WhoseTurn = "W"

 GameOver = False

 SampleGame = input("Do you want to play the sample game (enter Y for Yes)? ")

 if ord(SampleGame) >= 97 and ord(SampleGame) <= 122:

 SampleGame = chr(ord(SampleGame) - 32)

 InitialiseBoard(Board, SampleGame)

 while not(GameOver):

 DisplayBoard(Board)

 DisplayWhoseTurnItIs(WhoseTurn)

 MoveIsLegal = False

 while not(MoveIsLegal):

 StartSquare, FinishSquare = GetMove(StartSquare, FinishSquare)

 StartRank = StartSquare % 10

 StartFile = StartSquare // 10

 FinishRank = FinishSquare % 10

 FinishFile = FinishSquare // 10

 MoveIsLegal = CheckMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn)

 if not(MoveIsLegal):

 print("That is not a legal move - please try again")

 GameOver = CheckIfGameWillBeWon(Board, FinishRank, FinishFile)

 MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn)

 if GameOver:

 DisplayWinner(WhoseTurn)

 if WhoseTurn == "W":

 WhoseTurn = "B"

 else:

 WhoseTurn = "W"

 PlayAgain = input("Do you want to play again (enter Y for Yes)? ")

 if ord(PlayAgain) >= 97 and ord(PlayAgain) <= 122:

 PlayAgain = chr(ord(PlayAgain) - 32)