

**Programming Tasks (Mark Scheme)**

# Task 1 (2 marks)

## Coding:

* Create a new method CreateCustomPlayers which allows the user to enter in two player names and won’t continue until the names are different. **[1 mark]**

### Example Solution

Modify constructor in Dastan:

      self.\_MoveOptionOffer = []

**#CHANGE**

       self.\_\_CreateCustomPlayers()

**#END CHANGE**

      self.\_\_CreateMoveOptions()

New private method:

**#CODE ADDED**

**def** \_\_CreateCustomPlayers(self):

        p1 = input("Enter name for Player One: ")

        self.\_Players.append(Player(p1, 1))

        p2 = input("Enter name for Player Two: ")

        while p1==p2:

            p2 = input("Name matches Player One, enter a different name for Player Two: ")

        self.\_Players.append(Player("Player Two", -1))

**#END ADDITION**

## Testing:

Display an appropriate error message if the user enters in two matching names. Correctly identify player one with a custom name. **[1 mark]**

Enter name for Player One: Tom

Enter name for Player Two: Tom

Name matches Player One, enter a different name for Player Two: Tom

Name matches Player One, enter a different name for Player Two: Victoria

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Tom

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Tom

Choose move option to use from queue (1 to 3) or 9 to take the offer:

# Task 2 (4 marks)

## Coding:

* Adding a new MoveOptionOffer to the CreateMoveOptionOffer method, a new option in the CreateMoveOption method, and adding the move option to both players in the Player list with the Direction parameter set correctly. **[1 mark]**
* Adding a ‘faris’ to the CreateMoveOption method, a new option which calls the CreateFarisMoveOption method. **[1 mark]**
* Create a new method CreateFarisMoveOption which correctly uses the Direction parameter to identify all of the possible positions for the Faris move. **[1 mark]**

### Example Solution

Changes to CreateMoveOptionOffer:

**def** \_\_CreateMoveOptionOffer(self):

        self.\_MoveOptionOffer.append("faris") **#LINE ADDED**

Changes to CreateMoveOption:

**def** \_\_CreateMoveOption(self, Name, Direction):

        if Name == "chowkidar":

            return self.\_\_CreateChowkidarMoveOption(Direction)

**#CODE ADDED**

        elif Name == "faris":

            return self.\_\_CreateFarisMoveOption(Direction)

**#END ADDITION**

Code for CreateFarisMoveOption:

**#CODE ADDED**

**def** \_\_CreateFarisMoveOption(self, Direction):

        NewMoveOption = MoveOption("faris")

        NewMove = Move(1 \* Direction, 2 \* Direction)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(1 \* Direction, -2 \* Direction)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(-1 \* Direction, 2 \* Direction)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(-1 \* Direction, -2 \* Direction)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(2 \* Direction, 1 \* Direction)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(2 \* Direction, -1 \* Direction)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(-2 \* Direction, 1 \* Direction)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(-2 \* Direction, -1 \* Direction)

        NewMoveOption.AddToPossibleMoves(NewMove)

        return NewMoveOption

**#END ADDITION**

Changes to CreateMoveOptions:

**def** \_\_CreateMoveOptions(self):

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("ryott", 1))

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("faris", 1)) **#LINE ADDED**

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("chowkidar", 1))

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("cuirassier", 1))

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("faujdar", 1))

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("jazair", 1))

        self.\_Players[1].AddToMoveOptionQueue(self.\_\_CreateMoveOption("ryott", -1))

        self.\_Players[1].AddToMoveOptionQueue(self.\_\_CreateMoveOption("faris", -1)) **#LINE ADDED**

## Testing:

Displaying the Faris move option correctly in the player one queue and moving a player one piece appropriately for a legal Faris move. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: faris

Player One

Score: 100

Move option queue: 1. ryott 2. faris 3. chowkidar 4. cuirassier 5. faujdar 6. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 43

New score: 101

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | | | | | |

4 | | | !| | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: faris

Player Two

Score: 100

Move option queue: 1. ryott 2. faris 3. chowkidar 4. jazair 5. faujdar 6. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 55

Enter the square to move to (row number followed by column number): 43

New score: 102

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | | | | | |

4 | | | "| | | |

5 | | "| "| "| | |

6 | | | |k2| | |

-------------------

Move option offer: faris

Player One

Score: 101

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair 6. faris

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer:

# Task 3 (4 marks)

## Coding:

* Adding a new MoveOptionOffer to the CreateMoveOptionOffer method, a new option in the CreateMoveOption method, and adding the move option to both players in the Player list with the Direction parameter set correctly. **[1 mark]**
* Adding a Sarukh to the CreateMoveOption method, a new option which calls the CreateSarukhMoveOption method. **[1 mark]**
* Create a new method CreateSarukhMoveOption which correctly uses the Direction parameter to identify all the possible positions for the Sarukh move. **[1 mark]**

### Example Solution

Changes to CreateMoveOptionOffer:

**def** \_\_CreateMoveOptionOffer(self):

        self.\_MoveOptionOffer.append("sarukh") **#LINE ADDED**

Changes to CreateMoveOption:

**def** \_\_CreateMoveOption(self, Name, Direction):

        if Name == "chowkidar":

            return self.\_\_CreateChowkidarMoveOption(Direction)

**#CODE ADDED**

        elif Name == "sarukh":

            return self.\_\_CreateSarukhMoveOption(Direction)

**#END ADDITION**

Changes to CreateMoveOptions:

**def** \_\_CreateMoveOptions(self):

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("ryott", 1))

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("sarukh", 1)) **#LINE ADDED**

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("chowkidar", 1))

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("cuirassier", 1))

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("faujdar", 1))

        self.\_Players[0].AddToMoveOptionQueue(self.\_\_CreateMoveOption("jazair", 1))

        self.\_Players[1].AddToMoveOptionQueue(self.\_\_CreateMoveOption("ryott", -1))

        self.\_Players[1].AddToMoveOptionQueue(self.\_\_CreateMoveOption("sarukh", -1)) **#LINE ADDED**

Code for CreateSarukhMoveOption:

**#CODE ADDED**

**def** \_\_CreateSarukhMoveOption(self, Direction):

        NewMoveOption = MoveOption("sarukh")

        NewMove = Move(0, -1)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(1 \* Direction, -1)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(2 \* Direction, 0)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(0, 1)

        NewMoveOption.AddToPossibleMoves(NewMove)

        NewMove = Move(1 \* Direction, 1)

        NewMoveOption.AddToPossibleMoves(NewMove)

        return NewMoveOption

**#END ADDITION**

## Testing:

Displaying the Sarukh move option correctly in the player one queue and moving a player one piece appropriately for a legal Sarukh move. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: sarukh

Player One

Score: 100

Move option queue: 1. ryott 2. sarukh 3. chowkidar 4. cuirassier 5. faujdar 6. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 42

New score: 101

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | | | | | |

4 | | !| | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: sarukh

Player Two

Score: 100

Move option queue: 1. ryott 2. sarukh 3. chowkidar 4. jazair 5. faujdar 6. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 52

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 52

Enter the square to move to (row number followed by column number): 51

New score: 101

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | | | | | |

4 | | !| | | | |

5 | "| | "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: sarukh

Player One

Score: 101

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair 6. sarukh

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer:

# Task 4 (5 marks)

## Coding:

* Change PlayGame to randomly award a Wafr to the current player and if one has been awarded, advise the player that they can select any queue position without cost. **[1 mark]**
* Change PlayGame so that if a move is legal and a Wafr has been awarded to the player, the move does not incur a points cost to the player. **[1 mark]**
* Create a new method AwardWafr in the Dastan class which has a 25% chance of returning true. **[1 mark]**
* Adding the WafrAwarded attribute to Player with get/set methods for WafrAwarded. **[1 mark]**

### Example Solution

Changes to Player:

**class** Player:

**def** \_\_init\_\_(self, N, D):

        self.\_\_WafrAwarded = False *#LINE ADDDED*

**#CODE ADDED**

**def** GetWafrAwarded(self):

        return self.\_\_WafrAwarded

**def** SetWafrAwarded(self):

        self.\_\_WafrAwarded = True

**#CODE ADDED**

Code for AwardWafr:

**#CODE ADDED**

**def** \_\_AwardWafr(self):

        return random.randint(1, 4) == 1

**#END ADDITION**

Changes to PlayGame:

**def** PlayGame(self):

        GameOver = False

        while not GameOver:

            self.\_\_DisplayState()

            SquareIsValid = False

            Choice = 0

**# CODE CHANGE**

            Wafr = False

            if self.\_\_AwardWafr() and not self.\_CurrentPlayer.GetWafrAwarded():

                print("You have been awarded a Wafr, you can select any move from your queue for free this turn.")

                self.\_CurrentPlayer.SetWafrAwarded()

                Wafr = True

            if Wafr:

                while Choice < 1 or Choice > 5:

                    Choice = int(input("Choose move option to use from queue (1 to 5): "))

            else:

                while Choice < 1 or Choice > 3:

                    Choice = int(input("Choose move option to use from queue (1 to 3) or 9 to take the offer: "))

                    if Choice == 9:

                        self.\_\_UseMoveOptionOffer()

                        self.\_\_DisplayState()

**#END CHANGE**

            if MoveLegal:

                PointsForPieceCapture = self.\_\_CalculatePieceCapturePoints(FinishSquareReference)

**#CODE CHANGE**

                if not Wafr:

                    self.\_CurrentPlayer.ChangeScore(-(Choice + (2 \* (Choice - 1))))

**#END CHANGE**

                self.\_CurrentPlayer.UpdateQueueAfterMove(Choice)

## Testing:

Show a player being awarded a Wafr and selecting a move from position 4 or 5 in the move option queue without incurring a cost. **[1 mark]**

Move option offer: jazair

Player Two

Score: 104

Move option queue: 1. chowkidar 2. jazair 3. faujdar 4. cuirassier 5. ryott

Turn: Player Two

You have been awarded a Wafr, you can select any move from your queue for free this turn.

Choose move option to use from queue (1 to 5): 5

Enter the square containing the piece to move (row number followed by column number): 42

Enter the square to move to (row number followed by column number): 32

New score: 109

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | "| | | | |

4 | | | | | | |

5 | | | "| "| "| |

6 | | | |k2| | |

-------------------

# Task 5 (5 marks)

## Coding:

* Change PlayGame to give new menu option 8 and reduction of player score by 5 if option 8 is selected. **[1 mark]**
* Adding the Opponent variable (or similar) to PlayGame and correctly assigning it to the player whose turn it is not. **[1 mark]**
* Correctly printing out the opponent’s queue. **[1 mark]**
* Creation of GetJustQueue which calls the GetQueueAsString method for the private attribute Queue in Player. **[1 mark]**

### Example Solution

Changes to PlayGame:

            while Choice < 1 or Choice > 3:

                Choice = int(input("Choose move option to use from queue (1 to 3), 8 to see your opponent's queue or 9 to take the offer: "))

                if Choice == 9:

                    self.\_\_UseMoveOptionOffer()

                    self.\_\_DisplayState()

**#CODE ADDED**

                elif Choice == 8:

                    Opponent = self.\_Players[1] if self.\_CurrentPlayer.SameAs(self.\_Players[0])  
 else self.\_Players[0]

                    print(Opponent.GetJustQueue())

                    self.\_CurrentPlayer.ChangeScore(-5)

                    print("New score: " + str(self.\_CurrentPlayer.GetScore()) + "\n")

**#END ADDITION**

Changes to Player:

**#CODE ADDED**

**def** GetJustQueue(self):

        return self.\_\_Queue.GetQueueAsString()

**#END ADDITION**

## Testing:

Display new menu option. Player one to select option 8 to view Player two’s queue. **[1 mark]**

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3), 8 to see your opponent's queue or 9 to take the offer: 8

1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

New score: 95

Choose move option to use from queue (1 to 3), 8 to see your opponent's queue or 9 to take the offer:

# Task 6 (5 marks)

## Coding:

* Create a new method GetValidInt which returns true if the user enters in a valid integer. It should print out a suitable message and force the user to retry until they have entered a valid integer. **[1 mark]**
* Change PlayGame to use the GetValidInt method on the main game menu to only allow valid integer input for the move queue choice. **[1 mark]**
* Change GetSquareReference to use the GetValidInt method for choosing a position square to print out a suitable error message only allow valid integer input. **[1 mark]**
* Change UseMoveOptionOffer to use the GetValidInt method for choosing which move option to replace. Only allow valid integer input and include a range of 1 to 5 to prevent an invalid queue position being selected. This should include a valid error message and a prompt to re-enter until it is valid. **[1 mark]**

**A:** passing of the prompt to GetValidInt() instead, but do not award if the line spacing is wrong.

### Example Solution

Code for GetValidInt:

**#CODE ADDED**

**def** \_\_GetValidInt(self):

        valid = False

        number = 0

        while not valid:

            userInput = input()

            try:

                number = int(userInput)

                valid = True

            except:

                print("Invalid input, you must enter an integer, please try again: ", end="")

        return number

**#END ADDITION**

Changes to PlayGame:

            while Choice < 1 or Choice > 3:

**#CODE CHANGE**

                print("Choose move option to use from queue (1 to 3) or 9 to take the offer: ",end="")

                Choice = self.\_\_GetValidInt()

**#END CHANGE**

                if Choice == 9:

                    self.\_\_UseMoveOptionOffer()

                    self.\_\_DisplayState()

            while not SquareIsValid:

                StartSquareReference = self.\_\_GetSquareReference("containing the piece to move")

                SquareIsValid = self.\_\_CheckSquareIsValid(StartSquareReference, True)

**#CODE ADDED**

                if not SquareIsValid:

                    print("You must enter a valid square.")

**#END ADDITION**

            SquareIsValid = False

            while not SquareIsValid:

                FinishSquareReference = self.\_\_GetSquareReference("to move to")

                SquareIsValid = self.\_\_CheckSquareIsValid(FinishSquareReference, False)

**#CODE ADDED**

                if not SquareIsValid:

                    print("You must enter a valid square.")

**#END ADDITION**

            MoveLegal = self.\_CurrentPlayer.CheckPlayerMove(Choice, StartSquareReference, FinishSquareReference)

Changes to GetSquareReference:

**def** \_\_GetSquareReference(self, Description):

**#CODE CHANGE**

        print("Enter the square " + Description + " (row number followed by column number): ",  
end="")

        SelectedSquare = self.\_\_GetValidInt()

**#END CHANGE**

        return SelectedSquare

Changes to UseMoveOptionOffer:

**def** \_\_UseMoveOptionOffer(self):

**#CODE CHANGE**

        ReplaceChoice = -1

        print("Choose the move option from your queue to replace (1 to 5): ",end="")

        while ReplaceChoice <1 or ReplaceChoice>5:

            ReplaceChoice=self.\_\_GetValidInt()

            if ReplaceChoice < 1 or ReplaceChoice > 5:

                print("You must enter a number from 1-5.")

                print("Please re-enter your selection: ", end="")

**#END CHANGE**

        self.\_CurrentPlayer.UpdateMoveOptionQueueWithOffer(ReplaceChoice - 1, self.\_\_CreateMoveOption(self.\_MoveOptionOffer[self.\_MoveOptionOfferPosition], self.\_CurrentPlayer.GetDirection()))

        self.\_CurrentPlayer.ChangeScore(-(10 - (ReplaceChoice \* 2)))

        self.\_MoveOptionOfferPosition = random.randint(0, 4)

## Testing:

Display an appropriate error message if the user enters in non-valid inputs for the main game menu and as a position to place MoveOptionOffer in the queue. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: help

Invalid input, you must enter an integer, please try again: 1

Enter the square containing the piece to move (row number followed by column number): 19

You must enter a valid square.

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 32

New score: 104

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | !| | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 9

Choose the move option from your queue to replace (1 to 5): 8

You must enter a number from 1-5.

Please re-enter your selection:

# Task 7 (5 marks)

## Coding:

* Adding the ChoiceOptionsLeft attribute to Player with getter method. Initialising attribute to 3. **[1 mark]**
* Create a new method DecreaseChoiceOptionsLeft in Player which decrements ChoiceOptionsLeft AND advise the player how many move options they have left. **[1 mark]**
* Change PlayGame to test if the player has used all their offer options and, if so, do not display option 9. **[1 mark]**
* Change UserMoveOptionOffer to call DecreaseChoiceOptionsLeft for the current player when they choose a move option from the menu. **[1 mark]**

### Example Solution

Changes to Player:

**class** Player:

**def** \_\_init\_\_(self, N, D):

        self.\_\_Score = 100

        self.\_\_Name = N

        self.\_\_Direction = D

        self.\_\_Queue = MoveOptionQueue()

**#CODE ADDED**

        self.\_\_ChoiceOptionsLeft = 3

**def** GetChoiceOptionsLeft(self):

        return self.\_\_ChoiceOptionsLeft

**def** DecreaseChoiceOptionsLeft(self):

        self.\_\_ChoiceOptionsLeft -= 1

**#END ADDITION**

Changes to PlayGame:

            while Choice < 1 or Choice > 3:

**#CODE ADDED**

                if self.\_CurrentPlayer.GetChoiceOptionsLeft() > 0:

                    Choice = int(input("Choose move option to use from queue (1 to 3) or 9 to take the offer: "))

                else:

**#END ADDITION**

                    Choice = int(input("Choose move option to use from queue (1 to 3): "))

                if self.\_CurrentPlayer.GetChoiceOptionsLeft() > 0 and Choice == 9:

                    self.\_\_UseMoveOptionOffer()

                    self.\_\_DisplayState()

Changes to UseMoveOptionOffer:

**def** \_\_UseMoveOptionOffer(self):

        ReplaceChoice = int(input("Choose the move option from your queue to replace   
(1 to 5): "))

        self.\_CurrentPlayer.UpdateMoveOptionQueueWithOffer(ReplaceChoice - 1, self.\_\_CreateMoveOption(self.\_MoveOptionOffer[self.\_MoveOptionOfferPosition], self.\_CurrentPlayer.GetDirection()))

        self.\_CurrentPlayer.ChangeScore(-(10 - (ReplaceChoice \* 2)))

        self.\_MoveOptionOfferPosition = random.randint(0, 4)

**#CODE ADDED**

        self.\_CurrentPlayer.DecreaseChoiceOptionsLeft()

print(f"You have {self.\_CurrentPlayer.GetChoiceOptionsLeft()} move option choices

left.")

**#END ADDITION**

## Testing:

Show player one selecting a move from the move option offer menu and decreasing the offer counter by 1. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 9

Choose the move option from your queue to replace (1 to 5): 1

You have 2 move option choices left.

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: ryott

Player One

Score: 92

Move option queue: 1. jazair 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 9

Choose the move option from your queue to replace (1 to 5): 2

You have 1 move option choices left.

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: faujdar

Player One

Score: 86

Move option queue: 1. jazair 2. ryott 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 9

Choose the move option from your queue to replace (1 to 5): 3

You have 0 move option choices left.

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: cuirassier

Player One

Score: 82

Move option queue: 1. jazair 2. ryott 3. faujdar 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3): 9

Choose move option to use from queue (1 to 3):

# Task 8 (5 marks)

## Coding:

* Writing the ResetQueueBackAfterUndo method which calls the ResetQueueBack method with one parameter (the same one it was passed - Position – but adjusted by -1 to make it an index into the Queue) and successfully pops the item from the end of the queue and returns it to its original position. **[1 mark]**
* Asking the player if they would like to undo after they have played their move and seen its effect. **[1 mark]**
* Correctly handling the undo to deduct 5 points and reset the board and queue. **[1 mark]**

### Example Solution

Changes to Player:

**#CODE ADDED**

**def** ResetQueueBackAfterUndo(self, Choice):

        self.\_\_Queue.ResetQueueBack(Choice - 1)

**#END ADDITION**

Code for ResetQueBack:

**#CODE ADDED**

**def** ResetQueueBack(self, Position):

        self.\_\_Queue.insert(Position,self.\_\_Queue.pop(len(self.\_\_Queue)-1))

**#END ADDITION**

Changes to PlayGame:

            MoveLegal = self.\_CurrentPlayer.CheckPlayerMove(Choice, StartSquareReference, FinishSquareReference)

**#CODE CHANGE**

            undo = "n"

            if MoveLegal:

                PreviousScore = self.\_CurrentPlayer.GetScore()

                PointsForPieceCapture = self.\_\_CalculatePieceCapturePoints(FinishSquareReference)

                self.\_CurrentPlayer.ChangeScore(-(Choice + (2 \* (Choice - 1))))

                self.\_CurrentPlayer.UpdateQueueAfterMove(Choice)

                self.\_\_UpdateBoard(StartSquareReference, FinishSquareReference)

                self.\_\_UpdatePlayerScore(PointsForPieceCapture)

                print("New score: " + str(self.\_CurrentPlayer.GetScore()) + "\n")

                self.\_\_DisplayState()

                undo = input("Would you like to undo your move (y/n)? ")

                if undo.strip().lower() == "y":

                    self.\_\_UpdateBoard(FinishSquareReference,StartSquareReference)

                    self.\_CurrentPlayer.ResetQueueBackAfterUndo(Choice)

                    self.\_CurrentPlayer.ChangeScore(PreviousScore-self.\_CurrentPlayer.GetScore()-5)

                else:

                    undo = "n"

            if undo == "n":

                if self.\_CurrentPlayer.SameAs(self.\_Players[0]):

                    self.\_CurrentPlayer = self.\_Players[1]

                else:

                    self.\_CurrentPlayer = self.\_Players[0]

**#END CHANGE**

            GameOver = self.\_\_CheckIfGameOver()

## Testing:

* Showing that a move can be undone and that 5 points are deducted. **[1 mark]**
* Showing that the same player can still play their turn and that the game can continue normally after an undo.   
  **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 31

New score: 101

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | !| | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 101

Move option queue: 1. ryott 2. cuirassier 3. faujdar 4. jazair 5. chowkidar

Turn: Player One

Would you like to undo your move (y/n)? y

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 95

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 32

New score: 99

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | !| | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 99

Move option queue: 1. chowkidar 2. cuirassier 3. faujdar 4. jazair 5. ryott

Turn: Player One

Would you like to undo your move (y/n)? n

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | !| | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer:

# Task 9 (7 marks)

## Coding:

* CreateMoveOptionOffer method has been modified to the append “sahm” as a move option offer and correctly dealing with the Name parameter of “sahm” in the CreateMoveOption method. **[1 mark]**
* Making the Sahm the move option for both players on their first turn. **[1 mark]**
* Correctly creating the SahmUsed attribute with getter/setter methods. **[1 mark]**
* Only allowing a player to fire a single Sahm in a game. **[1 mark]**
* Correctly removing all the pieces in the Sahm’s line of fire from the board (except any in a Kotla) in the CalculateSahmMove method. **[1 mark]**
* Correctly awarding points for all removed/destroyed pieces (even if a piece was removed in error) in the CalculateSahmMove method. **[1 mark]**

### Example Solution

Changes to CreateMoveOptionOffer:

**def** \_\_CreateMoveOptionOffer(self):

        self.\_MoveOptionOffer.append("sahm") **#LINE ADDED**

        self.\_MoveOptionOffer.append("jazair")

Code for CreateSahmMoveOption:

**#CODE ADDED**

**def** \_\_CreateSahmMoveOption(self, Direction):

        NewMoveOption = MoveOption("sahm")

        NewMove = Move(0, 0)

        NewMoveOption.AddToPossibleMoves(NewMove)

        return NewMoveOption

**#END ADDITION**

Changes to CreateMoveOption:

**def** \_\_CreateMoveOption(self, Name, Direction):

        if Name == "chowkidar":

            return self.\_\_CreateChowkidarMoveOption(Direction)

**#CODE ADDED**

        elif Name == "sahm":

            return self.\_\_CreateSahmMoveOption(Direction)

**#END ADDITION**

        elif Name == "ryott":

Changes to Player:

        self.\_\_Queue = MoveOptionQueue()

**#CODE ADDED**

        self.\_\_SahmtUsed = False

**def** GetSahmUsed(self):

        return self.\_\_SahmUsed

**def** SetSahmUsed(self):

        self.\_\_SahmUsed = True

**def** ChoiceIsSahm(self, Choice):

        return self.\_\_Queue.GetMoveOptionInPosition(Choice-1).GetName() == "sahm"

**#END ADDITION**

Changes to PlayGame:

                SquareIsValid = self.\_\_CheckSquareIsValid(StartSquareReference, True)

**#CODE CHANGE**

            if self.\_CurrentPlayer.ChoiceIsSahm(Choice):

                if self.\_CurrentPlayer.GetSahmUsed():

                    print("You have already used your Sahm!")

                else:

                    self.\_CurrentPlayer.SetSahmUsed()

                    PointsForPieceCapture = self.\_\_CalculateSahmMove(StartSquareReference)

                    self.\_CurrentPlayer.ChangeScore(-(Choice + (2 \* (Choice - 1))))

                    self.\_CurrentPlayer.UpdateQueueAfterMove(Choice)

                    self.\_\_UpdatePlayerScore(PointsForPieceCapture)

                    print("New score: " + str(self.\_CurrentPlayer.GetScore()) + "\n")

            else:

                SquareIsValid = False

                while not SquareIsValid:

                    FinishSquareReference = self.\_\_GetSquareReference("to move to")

                    SquareIsValid = self.\_\_CheckSquareIsValid(FinishSquareReference, False)

                MoveLegal = self.\_CurrentPlayer.CheckPlayerMove(Choice, StartSquareReference, FinishSquareReference)

                if MoveLegal:

                    PointsForPieceCapture = self.\_\_CalculatePieceCapturePoints(FinishSquareReference)

                    self.\_CurrentPlayer.ChangeScore(-(Choice + (2 \* (Choice - 1))))

                    self.\_CurrentPlayer.UpdateQueueAfterMove(Choice)

                    self.\_\_UpdateBoard(StartSquareReference, FinishSquareReference)

                    self.\_\_UpdatePlayerScore(PointsForPieceCapture)

                    print("New score: " + str(self.\_CurrentPlayer.GetScore()) + "\n")

**#END CHANGE**

            if self.\_CurrentPlayer.SameAs(self.\_Players[0]):

Code for CalculateSahmMove:

**#CODE ADDED**

**def** \_\_CalculateSahmMove(self, SquareReference):

        Row = SquareReference // 10

        Col = SquareReference % 10

        Score = 0

        Direction = self.\_CurrentPlayer.GetDirection()

        if Direction == 1:

            EndRow = 6

        else:

            EndRow = 1

        while Row != EndRow:

            Row += Direction

            BoardSquare = self.\_Board[self.\_\_GetIndexOfSquare(10\*Row+Col)]

            if BoardSquare.GetPieceInSquare() is not None and not BoardSquare.ContainsKotla():

                Score += BoardSquare.GetPieceInSquare().GetPointsIfCaptured()

                BoardSquare.RemovePiece()

        return Score

**#END ADDITION**

## Testing:

* Showing the board correctly after the Sahm has been fired (allow follow-through for pieces in the Kotla destroyed). The pieces on 23 and 33 must have been destroyed to award the mark. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: sahm

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 33

New score: 101

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | | !| | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: sahm

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 9

Choose the move option from your queue to replace (1 to 5): 1

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | | !| | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 92

Move option queue: 1. sahm 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 53

New score: 98

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | | !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 101

Move option queue: 1. ryott 2. cuirassier 3. faujdar 4. jazair 5. chowkidar

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: raaket

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 33

New score: 101

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | | !| | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: raaket

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 9

Choose the move option from your queue to replace (1 to 5): 1

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | | !| | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: ryott

Player Two

Score: 92

Move option queue: 1. raaket 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 53

New score: 98

# Task 10 (4 marks)

## Coding:

* Add option 7 to the menu to create a Kotla. **[1 mark]**
* Checking that the square in which the player wishes to create the Kotla is empty. **[1 mark]**
* Creating a Kotla of the correct type in the square and removing the piece. **[1 mark]**

### Example Solution

Changes to PlayGame:

            while Choice < 1 or (Choice > 3 and Choice != 7):

                Choice = int(input("Choose move option to use from queue (1 to 3), 7 to create a Kotla or 9 to take the offer: "))

                if Choice == 9:

                    self.\_\_UseMoveOptionOffer()

                    self.\_\_DisplayState()

            if Choice==7:

                while not SquareIsValid:

                    StartSquareReference = self.\_\_GetSquareReference("containing the piece to sacrifice for a new Kotla")

                    SquareIsValid = self.\_\_CheckSquareIsValid(StartSquareReference, True)

                if self.\_CurrentPlayer.SameAs(self.\_Players[0]):

                    self.\_Board[self.\_\_GetIndexOfSquare(StartSquareReference)] = Kotla(self.\_CurrentPlayer, "K")

                else:

                    self.\_Board[self.\_\_GetIndexOfSquare(StartSquareReference)] = Kotla(self.\_CurrentPlayer, "k")

                self.\_CurrentPlayer

            else:

                while not SquareIsValid:

                    StartSquareReference = self.\_\_GetSquareReference("containing the piece to move")

                    SquareIsValid = self.\_\_CheckSquareIsValid(StartSquareReference, True)

                SquareIsValid = False

                while not SquareIsValid:

## Testing:

Showing the creation of the new Kotla (even if the letter is wrong) and removal of the sacrificed piece. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3), 7 to create a Kotla or 9 to take the offer: 7

Enter the square containing the piece to sacrifice for a new Kotla (row number followed by column number): 52

Enter the square containing the piece to sacrifice for a new Kotla (row number followed by column number): 22

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | |K | !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3), 7 to create a Kotla or 9 to take the offer:

# Task 11 (9 marks)

## Coding:

* Adding option 6 to the menu which brings up a list of options to modify the queue. **[1 mark]**
* Option a correctly reverses the player’s queue in a method inside MoveOptionQueue. **[1 mark]**
* Option b correctly swaps queue with the opponent without breaking encapsulation by exposing additional attributes that shouldn’t be exposed. **[1 mark]**
* Option c correctly swaps the first and last elements of your queue. **[1 mark]**
* Option d correctly moves an element to the front of the queue and shuffles up the others towards the end. **[1 mark]**
* Option e quits and doesn’t cost any points but the other options all cost 3 points. **[1 mark]**

### Example Solution

Changes to PlayGame:

            while Choice < 1 or Choice > 3:

                Choice = int(input("Choose move option to use from queue (1 to 3), 6 to modify the queue or 9 to take the offer: "))

                if Choice == 6:

                    self.\_\_ModifyQueueOptions()

                elif Choice == 9:

                    self.\_\_UseMoveOptionOffer()

                    self.\_\_DisplayState()

Code for ModifyQueueOptions:

**def** \_\_ModifyQueueOptions(self):

        print("You have the following options to modify your queue:")

        print("a) Reverse your queue")

        print("b) Swap queues with your opponent")

        print("c) Swap the first and last move options in your queue")

        print("d) Move an option of your choice to position 1 in the queue")

        print("e) Don't modify the queue, let me play my move!")

        option = input("Enter your choice (a-e)").lower()

        while option not in ["a","b","c","d","e"]:

            print("You must choose a letter from a to e: ", end="")

            option = input().lower()

        if option == "e":

            return

        elif option == "a":

            self.\_CurrentPlayer.ReverseQueue()

        elif option == "b":

            p1Queue = self.\_Players[0].ReplaceQueue(None)

            p2Queue = self.\_Players[1].ReplaceQueue(p1Queue)

            self.\_Players[0].ReplaceQueue(p2Queue)

        elif option == "c":

            self.\_CurrentPlayer.SwapFirstAndLast()

        elif option == "d":

            self.\_CurrentPlayer.MoveItemToFront(input("Enter position of item to move to the front(2-5): "))

self.\_CurrentPlayer.ChangeScore(-3)

        self.\_\_DisplayState()

Changes to MoveOptionQueue:

**def** ReverseQueue(self):

        self.\_\_Queue.reverse()

**def** MoveItemToFront(self, Position):

        self.\_\_Queue.insert(0,self.\_\_Queue.pop(Position))

**def** SwapFirstAndLast(self):

        Last = self.\_\_Queue.pop()

        self.\_\_Queue.append(self.\_\_Queue.pop(0))

        self.\_\_Queue.insert(0,Last)

**#END ADDITION**

Changes to Player:

**#CODE ADDED**

**def** ReverseQueue(self):

        self.\_\_Queue.ReverseQueue()

**def** ReplaceQueue(self, NewQueue):

        old = self.\_\_Queue

        self.\_\_Queue = NewQueue

        return old

**def** SwapFirstAndLast(self):

        self.\_\_Queue.SwapFirstAndLast()

**def** MoveItemToFront(self, ItemNumber):

        self.\_\_Queue.MoveItemToFront(ItemNumber-1)

**#END ADDITION**

## Testing:

* Showing at least one of options a–d working. **[1 mark]**
* Showing the remaining three options working. **[1 mark]**
* Showing option e and the scoring working correctly. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3), 6 to modify the queue or 9 to take the offer: 6

You have the following options to modify your queue:

a) Reverse your queue

b) Swap queues with your opponent

c) Swap the first and last move options in your queue

d) Move an option of your choice to position 1 in the queue

e) Don't modify the queue, let me play my move!

Enter your choice (a-e): a

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 97

Move option queue: 1. jazair 2. faujdar 3. cuirassier 4. chowkidar 5. ryott

Turn: Player One

Choose move option to use from queue (1 to 3), 6 to modify the queue or 9 to take the offer: 6

You have the following options to modify your queue:

a) Reverse your queue

b) Swap queues with your opponent

c) Swap the first and last move options in your queue

d) Move an option of your choice to position 1 in the queue

e) Don't modify the queue, let me play my move!

Enter your choice (a-e): b

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 94

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player One

Choose move option to use from queue (1 to 3), 6 to modify the queue or 9 to take the offer: 6

You have the following options to modify your queue:

a) Reverse your queue

b) Swap queues with your opponent

c) Swap the first and last move options in your queue

d) Move an option of your choice to position 1 in the queue

e) Don't modify the queue, let me play my move!

Enter your choice (a-e): c

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 91

Move option queue: 1. cuirassier 2. chowkidar 3. jazair 4. faujdar 5. ryott

Turn: Player One

Choose move option to use from queue (1 to 3), 6 to modify the queue or 9 to take the offer: 6

You have the following options to modify your queue:

a) Reverse your queue

b) Swap queues with your opponent

c) Swap the first and last move options in your queue

d) Move an option of your choice to position 1 in the queue

e) Don't modify the queue, let me play my move!

Enter your choice (a-e): d

Enter position of item to move to the front(2-5): 4

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 88

Move option queue: 1. faujdar 2. cuirassier 3. chowkidar 4. jazair 5. ryott

Turn: Player One

Choose move option to use from queue (1 to 3), 6 to modify the queue or 9 to take the offer: 6

You have the following options to modify your queue:

a) Reverse your queue

b) Swap queues with your opponent

c) Swap the first and last move options in your queue

d) Move an option of your choice to position 1 in the queue

e) Don't modify the queue, let me play my move!

Enter your choice (a-e): e

Choose move option to use from queue (1 to 3), 6 to modify the queue or 9 to take the offer:

# Task 12 (7 marks)

## Coding:

* Creating and storing the number of pieces correctly in the new protected attribute NoOfPieces. **[1 mark]**
* Adding a call to CheckReincarnation in the correct place. **[1 mark]**
* Creating CountNormalPieces to correctly return the number of pieces excluding the Mirza for the currently player. **[1 mark]**
* Correctly detecting when a piece reaches the opponent's back row. **[1 mark]**
* Having a condition to only allow reincarnation if the player has fewer pieces than they started with. **[1 mark]**
* Correctly handling the reincarnation on the player’s own back row and checking that the square is empty. **[1 mark]**

### Example Solution

Changes to Dastan:

**def** \_\_init\_\_(self, R, C, NoOfPieces):

        self.\_NoOfPieces = NoOfPieces **#LINE ADDED**

Changes to PlayGame:

            if MoveLegal:

                self.\_\_CheckReincarnation(FinishSquareReference) **#LINE ADDED**

Code for CountNormalPieces:

**#CODE ADDED**

**def** \_\_CountNormalPieces(self):

        Pieces = 0

        for S in self.\_Board:

            PieceInSquare = S.GetPieceInSquare()

            if PieceInSquare != None:

                if PieceInSquare.GetBelongsTo().SameAs(self.\_CurrentPlayer) and PieceInSquare.GetTypeOfPiece() == "piece":

                    Pieces += 1

        return Pieces

**def** \_\_CheckReincarnation(self, SquareReference):

        Row = SquareReference // 10

        Col = SquareReference % 10

        if self.\_CurrentPlayer.SameAs(self.\_Players[0]):

            if Row == self.\_NoOfRows and self.\_\_CountNormalPieces() < self.\_NoOfPieces:

                print("Congratulations, you have earned a reincarnation!")

                ReincarnationCol = int(input("Which column would you like your piece to be reincarnated on? "))

                while self.\_Board[self.\_\_GetIndexOfSquare(10+ReincarnationCol)]  
.GetPieceInSquare() != None:

                    print("The square must be empty.")

                    ReincarnationCol = int(input("Which column would you like your piece to be reincarnated on? "))

                self.\_Board[self.\_\_GetIndexOfSquare(10+ReincarnationCol)].SetPiece(Piece  
("piece", self.\_Players[0], 1, "!"))

        else:

            if Row == 1 and self.\_\_CountNormalPieces() < self.\_NoOfPieces:

                print("Congratulations, you have earned a reincarnation!")

                ReincarnationCol = int(input("Which column would you like your piece to be reincarnated on? "))

                while self.\_Board[self.\_\_GetIndexOfSquare(6\*10+ReincarnationCol)].  
GetPieceInSquare() != None:

                    print("The square must be empty.")

                    ReincarnationCol = int(input("Which column would you like your piece to be reincarnated on? "))

                self.\_Board[self.\_\_GetIndexOfSquare(6\*10+ReincarnationCol)].SetPiece(Piece("piece", self.\_Players[1], 1, '"'))

**#END ADDITION**

## Testing:

Correctly showing the moves as requested in the tests, specifically including the choice of the blocked square to reincarnate on and then the correct one. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | "| !| !| | | |

3 | | | | | | |

4 | | | | | | |

5 | !| "| "| "| | |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 51

Enter the square to move to (row number followed by column number): 61

Congratulations, you have earned a reincarnation!

Which column would you like your piece to be reincarnated on? 3

The square must be empty.

Which column would you like your piece to be reincarnated on? 4

New score: 104

1 2 3 4 5 6

-------------------

1 | | |K1| !| | |

2 | "| !| !| | | |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| | |

6 | !| | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 21

Enter the square to move to (row number followed by column number): 11

New score: 104

1 2 3 4 5 6

-------------------

1 | "| |K1| !| | |

2 | | !| !| | | |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| | |

6 | !| | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 104

Move option queue: 1. chowkidar 2. cuirassier 3. faujdar 4. jazair 5. ryott

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer:

# Task 13 (8 marks)

## Coding:

* Putting the Taziz in the correct place regardless of board size. **[1 mark]**
* Having a mechanism that correctly counts the number of turns that the Taziz has been camped for. **[1 mark]**
* Resetting the CampedTurns attribute if the square becomes empty or changes owner. **[1 mark]**
* Allowing the player to make a move that costs zero points when they have held the Taziz for two turns. **[1 mark]**
* Showing the correct A and a symbols when the Taziz is occupied by overriding SetPiece. **[1 mark]**
* Correctly resetting the symbol for the Taziz to ‘x’ when a player leaves by overriding RemovePiece. **[1 mark]**

### Example Solution

Changes to CreateBoard:

**#CODE ADDED**

elif Row == self.\_NoOfRows // 2 + 1 and Column == self.\_NoOfColumns // 2 + self.\_NoOfColumns % 2:

                    S = Taziz()

**#END ADDITION**

                else:

                    S = Square()

Changes to PlayGame:

**def** PlayGame(self):

        GameOver = False

        taziz = self.\_Board[self.\_\_GetIndexOfSquare((self.\_NoOfRows // 2 + 1)\*10 + (self.\_NoOfColumns // 2 + self.\_NoOfColumns % 2))] **#LINE ADDED**

        while not GameOver:

            if MoveLegal:

                PointsForPieceCapture = self.\_\_CalculatePieceCapturePoints(FinishSquareReference)

**#CODE CHANGE**

                self.\_CurrentPlayer.UpdateQueueAfterMove(Choice)

                self.\_\_UpdateBoard(StartSquareReference, FinishSquareReference)

                taziz.CheckCamp()

                if not taziz.GetCampedTwoTurns():

                    self.\_CurrentPlayer.ChangeScore(-(Choice + (2 \* (Choice - 1))))

**#END CHANGE**

                self.\_\_UpdatePlayerScore(PointsForPieceCapture)

Code for Taziz:

**#CODE ADDED**

**class** Taziz(Square):

**def** \_\_init\_\_(self):

        super(Taziz, self).\_\_init\_\_()

        self.\_Symbol = "x"

        self.\_CampedTurns = 0

**def** CheckCamp(self):

        if self.\_PieceInSquare != None:

          if (self.\_PieceInSquare.GetBelongsTo().GetDirection() == 1 and   
self.\_Symbol == "A") or (self.\_PieceInSquare.GetBelongsTo().GetDirection() == -1 and self.\_Symbol == "a"):

              self.\_CampedTurns += 1

**def** GetCampedTwoTurns(self):

        return self.\_CampedTurns >= 2

**def** SetPiece(self, P):

        super().SetPiece(P)

        if P.GetBelongsTo().GetDirection() == 1:

            self.\_Symbol = "A"

        else:

            self.\_Symbol = "a"

**def** RemovePiece(self):

        super().RemovePiece()

        self.\_Symbol = "x"

        self.\_CampedTurns = 0

**#END ADDITION**

## Testing:

* Show the Taziz being occupied and changing from x to A. **[1 mark]**
* Show player one getting a free move. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | |x | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 3

Enter the square containing the piece to move (row number followed by column number): 23

Enter the square to move to (row number followed by column number): 43

New score: 98

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| | !| !| |

3 | | | | | | |

4 | | |A!| | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 52

Enter the square to move to (row number followed by column number): 41

New score: 105

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| | !| !| |

3 | | | | | | |

4 | "| |A!| | | |

5 | | | "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 98

Move option queue: 1. ryott 2. chowkidar 3. faujdar 4. jazair 5. cuirassier

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 32

New score: 103

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | | !| !| |

3 | | !| | | | |

4 | "| |A!| | | |

5 | | | "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 105

Move option queue: 1. ryott 2. jazair 3. faujdar 4. cuirassier 5. chowkidar

Turn: Player Two

# Task 14 (10 marks)

## Coding:

* Using a method to track the Weather Event (this is the weatherEvent variable in the example below). **[1 mark]**
* Having the countdown timer allow precisely two complete turns from when it is announced. **[1 mark]**
* Announcing to the players when the Weather Event started with a 2 turns warning. **[1 mark]**
* Destroying all pieces in the same column as the Weather Event when the timer expires. **[1 mark]**
* Destroying a Kotla in the Weather Event column when the timer expires. **[1 mark]**
* Correctly selecting a random empty square. **[1 mark]**
* Creating a weatherEvent class with GetWeatherLocation and SetWeatherLocation working. **[1 mark]**
* Implementing CountDownComplete so that it returns an appropriate value including specifically when the time expires. **[1 mark]**

### Example Solution

Changes to PlayGame:

**def** PlayGame(self):

        GameOver = False

        weatherEvent = None **#LINE ADDED**

        while not GameOver:

            self.\_\_DisplayState()

            SquareIsValid = False

            Choice = 0

            while Choice < 1 or Choice > 3:

                Choice = int(input("Choose move option to use from queue (1 to 3) or 9 to take the offer: "))

                if Choice == 9:

                    self.\_\_UseMoveOptionOffer()

                    self.\_\_DisplayState()

            while not SquareIsValid:

                StartSquareReference = self.\_\_GetSquareReference("containing the piece to move")

                SquareIsValid = self.\_\_CheckSquareIsValid(StartSquareReference, True)

            SquareIsValid = False

            while not SquareIsValid:

                FinishSquareReference = self.\_\_GetSquareReference("to move to")

                SquareIsValid = self.\_\_CheckSquareIsValid(FinishSquareReference, False)

**#CODE ADDED**

if weatherEvent==None:

                weatherEvent = self.\_\_WeatherEventOccurs()

            else:

                if weatherEvent.CountDownComplete():

                    ColToDestroy = weatherEvent.GetWeatherLocation()%10

                    for Row in range (1,self.\_NoOfRows+1):

                        if self.\_Board[(self.\_\_GetIndexOfSquare(Row\*10+ColToDestroy))].  
GetPieceInSquare() != None:

                            self.\_Board[(self.\_\_GetIndexOfSquare(Row\*10+ColToDestroy))].  
RemovePiece()

if self.\_Board[(self.\_\_GetIndexOfSquare(Row\*10+ColToDestroy))].  
ContainsKotla():

                            self.\_Board[(self.\_\_GetIndexOfSquare(Row\*10+ColToDestroy))] = Square()

**#END ADDITION**

            MoveLegal = self.\_CurrentPlayer.CheckPlayerMove(Choice, StartSquareReference, FinishSquareReference)

Code for WeatherEventOccurs:

**#CODE ADDED**

**def** \_\_WeatherEventOccurs(self):

        weatherEvent = None

        if random.randint(1,2) == 1:

            squareFound = False

            while not squareFound:

                randomSquare = random.randint(1,self.\_NoOfRows)\*10 + random.randint(1,self.\_NoOfColumns)

                if self.\_Board[self.\_\_GetIndexOfSquare(randomSquare)].GetPieceInSquare() == None and not self.\_Board[self.\_\_GetIndexOfSquare(randomSquare)].  
ContainsKotla():

                    squareFound = True

            weatherEvent=WeatherEvent(randomSquare)

        return weatherEvent

**#END ADDITION**

Code for WeatherEvent:

**#CODE ADDED**

**class** WeatherEvent():

**def** \_\_init\_\_(self, SquareReference):

        self.\_\_SquareReference = SquareReference

        self.\_\_CountDownTimer = 3

        print("A weather event has occured at location",SquareReference)

        print("After two complete turns, all pieces on the same column will be destroyed.")

**def** SetWeatherLocation(self, SquareReference):

        self.\_\_SquareReference = SquareReference

**def** GetWeatherLocation(self):

        return self.\_\_SquareReference

**def** CountDownComplete(self):

        if self.\_\_CountDownTimer == 0:

            print("The weather event destroys all the pieces in column", self.\_\_SquareReference%10)

            return True

        else:

            self.\_\_CountDownTimer -= 1

            print("The weather event at location", self.\_\_SquareReference,"will occur", "after one more turn" if self.\_\_CountDownTimer>1 else "next turn")

            return False

**#END ADDITION**

## Testing:

* Having at least one piece owned by each player in the column where the weather event will occur so that both players will lose at least one piece each. **[1 mark]**
* Showing all pieces in the column destroyed. **[1 mark]**

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 22

Enter the square to move to (row number followed by column number): 32

A weather event has occured at location 61

After two complete turns, all pieces on the same column will be destroyed.

New score: 104

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | !| | | | |

4 | | | | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 52

Enter the square to move to (row number followed by column number): 51

The weather event at location 61 will occur after one more turn

New score: 104

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | !| !| !| |

3 | | !| | | | |

4 | | | | | | |

5 | "| | "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 104

Move option queue: 1. chowkidar 2. cuirassier 3. faujdar 4. jazair 5. ryott

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 32

Enter the square to move to (row number followed by column number): 21

The weather event at location 61 will occur next turn

New score: 108

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | !| | !| !| !| |

3 | | | | | | |

4 | | | | | | |

5 | "| | "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 104

Move option queue: 1. chowkidar 2. jazair 3. faujdar 4. cuirassier 5. ryott

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 53

Enter the square to move to (row number followed by column number): 42

The weather event at location 61 will occur next turn

New score: 108

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | !| | !| !| !| |

3 | | | | | | |

4 | | "| | | | |

5 | "| | | "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 108

Move option queue: 1. cuirassier 2. faujdar 3. jazair 4. ryott 5. chowkidar

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 23

Enter the square to move to (row number followed by column number): 43

The weather event destroys all the pieces in column 1

New score: 112

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | | | !| !| |

3 | | | | | | |

4 | | "| !| | | |

5 | | | | "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player Two

Score: 108

Move option queue: 1. jazair 2. faujdar 3. cuirassier 4. ryott 5. chowkidar

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer:

# Task 15 (15 marks)

## Coding:

* Creating a Barrier class that accepts the parameters Player and Symbol and sets the attributes correctly.   
  **[1 mark]**
* Creating ContainsBarrier that returns true for B or b and false otherwise. **[1 mark]**
* Modifying CheckSquareIsValid to return false if the end square contains a barrier. **[1 mark]**
* Creating CheckBarrierIsValid which checks that all the squares for the barrier are in bounds. **[1 mark]**
* Creating CheckBarrierIsValid which checks that all the squares for the barrier are empty. **[1 mark]**
* Creating PlaceBarrier with suitable input messages that calls CheckBarrierIsValid AND successfully creates a barrier 3 squares wide on the board. **[1 mark]**
* Modifying CreatePieces to make two calls to PlaceBarrier, one for each player. **[1 mark]**
* Creating CheckManhattanDistance and modifying PlayGame to call that instead of calling CheckPlayerMove for the line starting MoveLegal=. **[1 mark]**
* Inside CheckManhattanDistance, checking that the start and end squares are valid (preferably by calling CheckPlayerMove). **[1 mark]**
* Inside CheckManhattanDistance, iterating along the Row and Column and vice versa even if this doesn’t work and has only been attempted. **[1 mark]**
* Inside CheckManhattanDistance, correctly iterating along the Row and Column and vice versa for both players in all combinations of up, down, left and right (with and without vertical/horizontal movement) for both players in all possible move orientations (in the code below this was achieved by using the HorizontalDirection and VerticalDirection). **[1 mark]**

### Example Solution

Code for Barrier:

**#CODE ADDED**

**class** Barrier(Square):

**def** \_\_init\_\_(self, P, S):

        super(Barrier, self).\_\_init\_\_()

        self.\_BelongsTo = P

        self.\_Symbol = S

**#END ADDITION**

Changes to Square:

**#CODE ADDED**

**def** ContainsBarrier(self):

        if self.\_Symbol == "B" or self.\_Symbol == "b":

            return True

        else:

            return False

**#END ADDITION**

Changes to CheckSquareIsValid (Dastan class):

**def** \_\_CheckSquareIsValid(self, SquareReference, StartSquare):

        if not self.\_\_CheckSquareInBounds(SquareReference):

            return False

**#CODE ADDED**

        if self.\_Board[self.\_\_GetIndexOfSquare(SquareReference)].ContainsBarrier():

            return False

**#END ADDITION**

Code for CheckBarrierIsValid (Dastan class):

**def** \_\_CheckBarrierIsValid(self, BarrierCentre):

        BarrierIsValid = True

        BarrierRow = BarrierCentre // 10

        BarrierCol = BarrierCentre % 10

        for col in range(BarrierCol-1, BarrierCol+2):

            if not self.\_\_CheckSquareInBounds(BarrierRow\*10+col):

                BarrierIsValid = False

            else:

                if self.\_Board[self.\_\_GetIndexOfSquare(BarrierRow\*10+col)].ContainsKotla() or self.\_Board[self.\_\_GetIndexOfSquare(BarrierRow\*10+col)].ContainsBarrier() or self.\_Board[self.\_\_GetIndexOfSquare(BarrierRow\*10+col)].GetPieceInSquare() != None:

                    BarrierIsValid = False

        return BarrierIsValid

Code for PlaceBarrier (Dastan class):

**def** \_\_PlaceBarrier(self, P, S):

        BarrierValid = False

        while not BarrierValid:

            BarrierSquare = int(input("Which square would you like to be the centre of your horizontal, 3-square-wide barrier? "))

            if self.\_\_CheckBarrierIsValid(BarrierSquare):

                BarrierValid = True

            else:

                print("You must choose the centre of 3 empty squares on the board horizonally.")

        BarrierRow = BarrierSquare // 10

        BarrierCol = BarrierSquare % 10

        for col in range(BarrierCol-1, BarrierCol+2):

            self.\_Board[self.\_\_GetIndexOfSquare(BarrierRow\*10+col)] = Barrier(P,S)

**#END ADDITION**

Changes to CreatePieces (Dastan class):

**def** \_\_CreatePieces(self, NoOfPieces):

        for Count in range(1, NoOfPieces + 1):

            CurrentPiece = Piece("piece", self.\_Players[0], 1, "!")

            self.\_Board[self.\_\_GetIndexOfSquare(2 \* 10 + Count + 1)].SetPiece(CurrentPiece)

        CurrentPiece = Piece("mirza", self.\_Players[0], 5, "1")

        self.\_Board[self.\_\_GetIndexOfSquare(10 + self.\_NoOfColumns // 2)].SetPiece(CurrentPiece)

**#CODE CHANGE**

print("Player One, it's time to place your barrier")

        self.\_\_PlaceBarrier(self.\_Players[0],"B")

        for Count in range(1, NoOfPieces + 1):

            CurrentPiece = Piece("piece", self.\_Players[1], 1, '"')

            self.\_Board[self.\_\_GetIndexOfSquare((self.\_NoOfRows - 1) \* 10 + Count + 1)]  
.SetPiece(CurrentPiece)

        CurrentPiece = Piece("mirza", self.\_Players[1], 5, "2")

        self.\_Board[self.\_\_GetIndexOfSquare(self.\_NoOfRows \* 10 + (self.\_NoOfColumns // 2 + 1))].SetPiece(CurrentPiece)

        print("Player Two, it's time to place your barrier")

        self.\_\_PlaceBarrier(self.\_Players[1],"b")

**#END CHANGE**

Changes to PlayGame (Dastan class):

                SquareIsValid = self.\_\_CheckSquareIsValid(FinishSquareReference, False)

**#CODE CHANGE**

            MoveLegal = self.\_\_CheckManhattanDistance(Choice, StartSquareReference, FinishSquareReference)

**#END CHANGE**

            if MoveLegal:

Code for CheckManhattanDistance (Dastan class):

**#CODE ADDED**

**def** \_\_CheckManhattanDistance(self, Choice, StartSquare, EndSquare):

        if self.\_CurrentPlayer.CheckPlayerMove(Choice,StartSquare,EndSquare):

            HorizontalDirection = -1 if EndSquare%10<StartSquare%10 else 1

            VerticalDirection = -1 if EndSquare//10<StartSquare//10 else 1

            route1Valid = True

            for Row in range(StartSquare//10 + VerticalDirection, EndSquare//10 + VerticalDirection, VerticalDirection):

                if self.\_Board[self.\_\_GetIndexOfSquare(Row\*10+StartSquare%10)]  
.ContainsBarrier():

                    route1Valid = False

            for Col in range(StartSquare%10, EndSquare%10, HorizontalDirection):

                if self.\_Board[self.\_\_GetIndexOfSquare(EndSquare//10 \* 10 + Col)]  
.ContainsBarrier():

                    route1Valid = False

            route2Valid = True

            for Col in range(StartSquare%10 + HorizontalDirection, EndSquare%10 +   
HorizontalDirection, HorizontalDirection):

                if self.\_Board[self.\_\_GetIndexOfSquare(StartSquare//10+Col)]  
.ContainsBarrier():

                    route2Valid = False

            for Row in range(StartSquare//10, EndSquare//10, VerticalDirection):

                if self.\_Board[self.\_\_GetIndexOfSquare(Row \* 10 + EndSquare%10)].ContainsBarrier():

                    route2Valid = False

            return route1Valid or route2Valid

        return False

**#END ADDITION**

## Testing:

* Moving the piece correctly when only one route is valid. **[1 mark]**
* Not moving the piece for a cuirassier move when there is a barrier in the way. **[1 mark]**
* Not moving the piece when the end square is a barrier. **[1 mark]**
* Not moving the piece when there is a barrier in the way on both routes and the travel directions are both inverted (right to left and bottom to top). **[1 mark]**

Player One, it's time to place your barrier

Which square would you like to be the centre of your horizontal, 3-square-wide barrier? 34

Player Two, it's time to place your barrier

Which square would you like to be the centre of your horizontal, 3-square-wide barrier? 42

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | |B |B |B | |

4 |b |b |b | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: jazair

Player One

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 9

Choose the move option from your queue to replace (1 to 5): 1

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| !| !| |

3 | | |B |B |B | |

4 |b |b |b | | | |

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: ryott

Player One

Score: 92

Move option queue: 1. jazair 2. chowkidar 3. cuirassier 4. faujdar 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 24

Enter the square to move to (row number followed by column number): 46

New score: 96

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| | !| |

3 | | |B |B |B | |

4 |b |b |b | | | !|

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: ryott

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 3

Enter the square containing the piece to move (row number followed by column number): 53

Enter the square to move to (row number followed by column number): 31

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| | !| |

3 | | |B |B |B | |

4 |b |b |b | | | !|

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: ryott

Player One

Score: 96

Move option queue: 1. chowkidar 2. cuirassier 3. faujdar 4. jazair 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 2

Enter the square containing the piece to move (row number followed by column number): 25

Enter the square to move to (row number followed by column number): 45

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| | !| |

3 | | |B |B |B | |

4 |b |b |b | | | !|

5 | | "| "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: ryott

Player Two

Score: 100

Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 52

Enter the square to move to (row number followed by column number): 42

Enter the square to move to (row number followed by column number): 51

New score: 104

1 2 3 4 5 6

-------------------

1 | | |K1| | | |

2 | | !| !| | !| |

3 | | |B |B |B | |

4 |b |b |b | | | !|

5 | "| | "| "| "| |

6 | | | |k2| | |

-------------------

Move option offer: ryott

Player One

Score: 96

Move option queue: 1. chowkidar 2. cuirassier 3. faujdar 4. jazair 5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: