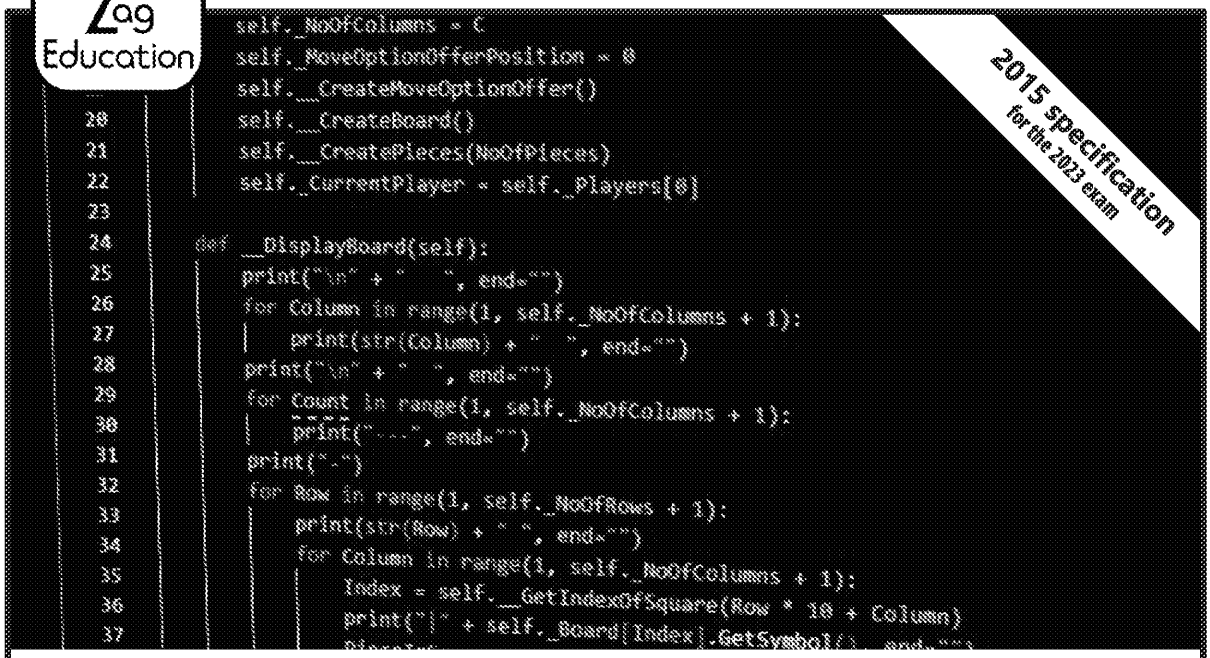
Zig Zag Education

Computer Science A Level | AQA | 7517



PAPER 1 EXAM RESOURCE PACK 2023

Dastan

for A Level AQA Computer Science

PYTHON³ EDITION

- DIGITAL RESOURCE -

This pack includes paper versions of the electronic files.

Go to **zzed.uk/ProductSupport** to download the electronic files.

POD 11730

zigzageducation.co.uk

Publish your own work... Write to a brief... Register at **publishmenow.co.uk**

Follow us on Twitter **@ZigZagComputing**

Photocopiable/digital resources may only be copied by the purchasing institution on a single site and for their own use

Contents

Product Support from ZigZag Education ii

Terms and Conditions of Use iii

Teacher's Introduction iv

Printouts of electronic resources (for reference)

- Code Breakdown (14 pages)
- UML Class Diagram – Complete (1 page)*
- Theory Questions: Write-on version (7 pages)
- Theory Questions: Non-write-on version (3 pages)
- Coding Tasks (19 pages)
- Additional Tasks (Extension) (6 pages)
- Theory Questions: Mark Scheme (3 pages)
- Programming Tasks: Mark Scheme (41 pages)
- Electronic Answer Document (3 pages)

** Note there are also electronic copies of the UML Diagrams ('Complete' & 'Activity' versions) which can be printed in A3, making them much more usable (especially when used as activities)*

Teacher's Introduction

This resource pack is designed to help you support your students taking the **A Level Computer Science Paper 1** exam. It is based on the ***Dastan*** preliminary material (Python[3]) – for examination summer 2023.

DIGITAL RESOURCE

Once you have downloaded the files for this resource via (**zzed.uk/ProductSupport**) you will have access to the following:

`Dastan`	this folder contains all of the content (PDF/DOCX) accessible via a HTML interface
`Passwords.txt`	for teacher use – this file contains all of the passwords for the protected PDFs (also listed below)

* PRINTED COPIES OF ALL THE MATERIALS IN THIS DIGITAL RESOURCE PACK ARE INCLUDED FOR REFERENCE.

Installation: Extract the files from the downloaded ZIP file and move the entire `Dastan` folder onto a network location that is accessible for students, and provide them with a shortcut to the index.html file. All content can be accessed from this page.

Passwords: All of the PDFs accessible via the *Solutions* web page are password-protected, so that students can only access them with your permission. Each password is a four-digit code, as follows:

`py02a-UML-Diagam-Complete.pdf`
`py06-TheoryQuestions-MS.pdf`
`py07-CodingTasks-MS.pdf`

The resource pack consists of the following:

① **Code Breakdown**

This document gives a detailed technical overview of the skeleton program, describing in detail each class and method in turn – including their purpose/function, parameters and return values.

> **Note:** although this section is intended to give extra support to teachers and students, it should in no way be seen as a substitute to a student exploring the code for themselves.

② **Class Diagrams**

Two UML Class Diagrams help students explore the skeleton program; there is a completed version and a partially-complete version which contains a total of 15 missing class and method names, data types, associations and access types (private/public) for students to fill in. The completed version is password-protected and accessible via the *Solutions* web page.

③ **Video**

A short video going over the *Dastan* game mechanics – intended as a visual aid to accompany the notes in the official AQA preliminary material.

④ **Written Questions**

Theory questions testing students' understanding of the skeleton program. These questions require access to the program, but no modifications need to be made to the program. Write-on (with answer lines) and non-write-on versions are available. Suggested answers are provided via the *Solutions* web page as a password-protected PDF.

⑤ **Coding Tasks**

Fifteen modification exercises put students' programming skills to the test. Example solutions with suggested mark schemes are provided via the *Solutions* web page as a password-protected PDF. Note that these are example solutions and you must use your discretion to award marks accordingly where there are valid alternative solutions.

An **Electronic Answer Document (EAD)** is provided should you wish students to use it for ③ and/or ④ above.

This resource is intended to supplement your teaching only. **Please read full disclaimer (p. iii) before using it.**

Dastan

Skeleton Code Breakdow

Important: Throughout this document and the Python code, methods are referred public. In this document, method names are written **without** leading underscores method names are written with leading underscores; a private method appears w start and a protected method with a single underscore.

Class: Dastan

Identifier / Data		Description
<<constructor>>		
Parameters	R : Int C : Int NoOfPieces : Int	Initialises the following protected attri • NoOfRows from parameter R • NoOfColumns from paramet • MoveOptionOfferPosition to
Return values	n/a	
		Instantiates two new Player objects – parameter of 1 and Player 2 with the and appends them both to the protec Assigns the element at position 0 of t (Player 1) to the protected attribute C Invokes the following methods: • CreateMoveOptions() – to add to each player. • CreateMoveOptionOffer() – to options to the move offer option • CreateBoard() – to create a sta • CreatePieces() – to add the sta to the board using the paramete
CalculatePieceCapturePoints (private)		
Parameters	FinishSquare Reference : Int	Uses the GetPieceInSquare method location from the FinishSquareRefe
Return values	Integer	If there is a piece at that location, the that piece is returned. If there is no pi returns 0.
CheckIfGameOver (private)		
Parameters	n/a	Iterates through the Board list checki
Return values	Boolean	If the square contains a piece, the me contains a Kotla, and the piece in the to the opponent of the player that own scenario, the player who owns the Mi opponent's Kotla. If this isn't the case confirm if the piece contains either a setting the Player1HasMirza and Pla appropriately. A negated logical AND of these two a players have lost their Mirza, the met returns false.

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

CheckSquareInBounds (private)		
Parameters	SquareReference : Int	Used as an error handling meth SquareReference parameter is playing board. The method initialises two local using DIV to split off the row an from the SquareReference par checks to confirm if Row is outs attribute NoOfRows and Col is the attribute NoOfColumns an If both are in range, the method
Return values	Boolean	

CheckSquareIsValid (private)		
Parameters	SquareReference : Int StartSquare : Boolean	Used to test if the SquareRefe Square choice. The StartSquare parameter is is being used to check when the location of a piece to move from otherwise it is passed as false to check when the player is sel piece to (a 'move to' check). The method firstly uses the Che method to confirm that the squa bounds of the board and return The method then gets the piece the SquareReference parame location and this is a 'move fro false because the player has s StartSquare parameter is true method instead returns true be a blank square. If there is a piece already at the the method checks to confirm i current player. If it does and thi method returns true. If this is a returns false because the playe onto one of their own pieces. If the piece does not belong to 'move from' check, the method player is trying to select an opp a 'move to' check, the method attempting to take an opponen
Return values	Boolean	

CreateBoard (private)		
Parameters	n/a	Uses nested iteration using the NoOfColumns attributes to po Player 1's Kotla is placed to the Player 2's Kotla is placed in the there is an even number of col NoOfRows attribute. The remaining locations are fill object.
Return values	n/a	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

CreateChowkidarMoveOption (private)		
Parameters	Direction : Int	Instantiates a new MoveOptio method uses the Direction pa Move objects – one for each v option. The first new Move parameter from starting location to finishin Move parameter is the numbe starting location to finishing loc to the starting location. A Direction of 1 moves down t Direction of -1 moves up the b Move object is added to the ch object which is then returned. See pre-release document for valid move positions (shown fr
Return values	NewMoveOption : MoveOption	

CreateCuirassierMoveOption (private)		
Parameters	Direction : Int	Instantiates a new MoveOptio method uses the Direction pa Move objects – one for each v option. The first new Move parameter from starting location to finishin Move parameter is the numbe starting location to finishing loc to the starting location. A Direction of 1 moves down t Direction of -1 moves up the b Move object is added to the cu object which is then returned. See pre-release document for valid move positions (shown fr
Return values	NewMoveOption : MoveOption	

CreateFaujdarMoveOption (private)		
Parameters	Direction : Int	Instantiates a new MoveOptio method uses the Direction pa Move objects – one for each v option. The first new Move parameter from starting location to finishin Move parameter is the numbe starting location to finishing loc to the starting location. A Direction of 1 moves down t Direction of -1 moves up the b Move object is added to the fa which is then returned. See pre-release document for valid move positions (shown fr
Return values	NewMoveOption : MoveOption	

INSPECTION COPY
COPYRIGHT PROTECTED

CreateJazairMoveOption (private)		
Parameters	Direction : Int	Instantiates a new MoveOptio method uses the Direction pa Move objects – one for each v option. The first new Move parameter from starting location to finishin Move parameter is the numbe starting location to finishing loc to the starting location. A Direction of 1 moves down t Direction of -1 moves up the b Move object is added to the ja which is then returned. See pre-release document for valid move positions (shown fr
Return values	NewMoveOption : MoveOption	

CreateRyottMoveOption (private)		
Parameters	Direction : Int	Instantiates a new MoveOptio method uses the Direction pa Move objects – one for each v option. The first new Move parameter from starting location to finishin Move parameter is the numbe starting location to finishing loc to the starting location. A Direction of 1 moves down t Direction of -1 moves up the b Move object is added to the ry which is then returned. See pre-release document for valid move positions (shown fr
Return values	NewMoveOption : MoveOption	

CreateMoveOption (private)		
Parameters	Name : String Direction : Int	Uses selection on the Name pa associated Create*****MoveO MoveOption from that method.
Return values	MoveOption	

CreateMoveOptionOffer (private)		
Parameters	n/a	Adds the first default MoveOpt starting attribute.
Return values	n/a	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

CreateMoveOptions (private)		
Parameters	n/a	Adds the five default MoveOpt MoveOptionQueue for each p This method calls the CreateM the move Name and Direction five defa[illegible]ove options, addi th[illegible]OptionQueue for Pla Players list.
Return values	n/a	

CreatePieces (private)		
Parameters	[illegible]ces : Int	Places the default playing piec onto the board. The method uses the NoOfPie many standard playing pieces Player 1 pieces on row 2 and P penultimate row. Pieces are gi which player they belong to, th their symbol on the board. Play symbol '!'. Player 2 pieces are using an escape character to c The method also places the Pl associated Kotlas by halving th work out the middle position in points value if captured of 5. P symbol of '1' and Player 2 Mirz
Return va	n/a	

DisplayBoard (private)		
Parameters	n/a	Iterates through the Board list The method works by using the • Iterate through to the num column number and a spa • Iterate through to the num sequence of hyphens. • Use nested iteration to pri for each square on the bo a piece in the square the printed, otherwise a blank • Print a final '|' symbol at th • Iterate through to the num sequence of hyphens follo
Return values	n/a	

DisplayFinalResult (private)		
Parameters	n/a	The winner of the game is the when this method is called. Th of b[illegible]ayers using the GetS If Player 1 has a higher score t uses the GetName() method to concatenated with 'is the winn Player 2 name concatenated w scores match, 'Draw!' is printe
Return values	n/a	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

DisplayState (private)		
Parameters	n/a	Used as part of the main menu method to display information a The method first calls the Disp board to the screen followed by for a pla[illegible]o choose if they w It th[illegible]ses the GetPlayerStat the score and move option que followed by the current player n
Return values	n/a	

GetIndexOf[illegible]a (private)		
Paramete	SquareReference : Int	Used to convert a SquareRefe the Board list for the associate The method initialises two loca using DIV to split off the row an from the SquareReference pa 1 is subtracted from both variat and then the Row is multiplied and added to the Col attribute a
Return values	Integer	

GetPointsForOccupancyByPlayer (private)		
Parameters	CurrentPlayer : Player	Used to calculate the total poin the CurrentPlayer. The method initialises an integ to 0 t[illegible]e[illegible]ates through the B which are occupied by the curr The GetPointsForOccupancy square in the Board list which method is overridden by the Ko Kotla belongs to current playe player Mirza or a current playe Kotla occupied by a current pla the opponent player. Points are totaled up in the Sc iteration progresses. This total is then returned.
Return values	ScoreAdjustment : Int	

GetSquareReference (private)		
Parameters	Description : String	Used to get a square reference The method uses the Descript an appropriate output to the us used f[illegible]r a start or finish s I[illegible]t from the user is casted w stored in a local integer variabl returned.
Return values	SelectedSquare : Int	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

PlayGame (public)	
Parameters	n/a
Return values	n/a

This method is the main game
the loop using the local Boolea

The method firstly displays the
includes the board and the cur
the curr ayer to choose a
m on queue or select 9
r e offer.

If the user selects option 9, the
UseMoveOptionOffer() to disp
and then displays the current g
loops until the user selects a va

The method then asks the use
StartSquareReference contai
like to move. Using the GetSq
CheckSquareIsValid() metho
the user gives a valid location.

The method then repeats this p
FinishSquareReference conta
player wants to move the piece
CheckPlayerMove() method t
StartStartReference and Fini
for the selected move choice.

If the move is legal, the metho

- Ca ulates any points if th
piece using the Calculate
and storing it in PointsFo
- Updates the player score
move option used from th
ChangeScore() method.
- Updates the player queue
MoveOption choice to the
UpdateQueueAfterMove
- Calls the UpdateBoard()
of pieces based on the St
FinishSquareReference.
- Calls the UpdatePlayerS
current player score with
- Prints the updated score f
screen.

This method does not deal with
not legal, it simply just ignores
player t ithout informing th

The method then checks which
swaps to the opposing player.
CheckIfGameOver() to check
their Mirza into the opponent K
been captured which stops the

After the main game playing lo
DisplayState() method to print
board and then calls the Displ
confirm which player has won.

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

UseMoveOptionOffer (private)		
Parameters	n/a	Used to place the move i into the current player mo The method asks the pla current offer move from t wi using any error h l integer variable Re uses the UpdateMoveO method on the CurrentP selected position move w the MoveOptionOffer lis the player score using the based on the position of t the player to replace. The method then update variable with a random n new move from the Mov
Return values	n/a	

UpdateBoard (private)		
Parameters	StartSquareReference : Int FinishSquareReference : Int	Performs the actual move on the board to another. The method uses the Re Board list index calculate StartSquareReference subsequently passed as nced to be placed at th om the FinishSquareR
Return values	n/a	

UpdatePlayerScore (private)		
Parameters	PointsForPieceCapture : Int	Calculates the change in which the player has just The method calls the GetPointsForOccupanc current player to create a Kotlas which are occupie added to the PointsForP contains the points for ar that move. The combined total is the player score using the C
Return va	n/a	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Class: Piece

Identifier / Data		Description
<<constructor>>		
Parameters	T : String B : Player P : Int S : String	Initialises the following protected • Type... iece from parame • ...gsTo from parameter • PointsIfCaptured from para • Symbol from parameter S
Return values	n/a	
GetBelongsTo (public)		
Parameters	n/a	Returns the value of the protecte
Return values	BelongsTo : Player	
GetPointsIfCaptured (public)		
Parameters	n/a	Returns the value of the protecte
Return values	PointsIfCaptured : Int	
GetSymbol (public)		
Parameters	n/a	Returns the value of the protecte
Return values	Symbol : String	
GetTypeOfPiece (public)		
Parameters	n/a	Returns the value of the protecte
Return values	TypeOfPiece : String	

Class: Square

Identifier / Data		Description
<<constructor>>		
Parameters	n/a	Initialises the following protected • PieceInSquare to null • BelongsTo to null • Symbol to ' '
Return values	n/a	
ContainsKotla (public) <<virtual>>		
Parameters	n/a	If the Symbol attribute is a 'K' o to confirm that there is a Kotla p returns false.
Return values	Boolean	
GetBelongsTo (public) <<virtual>>		
Parameters	n/a	Returns the value of the protecte
Return values	BelongsTo : Player	
GetPieceIn... (public) <<virtual>>		
Parameters	n/a	Returns the value of the protecte
Return values	PieceInSquare : Piece	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

GetPointsForOccupancy (public) <<virtual>>		
Parameters	CurrentPlayer : Player	Base class method for the Get method in the Kotla class to ov If the method was not overridd
Return values	Integer	
GetSymbol (public) <<virtual>>		
Parameters	n/a	R…n: …e value of the protec
Return values	Symbol : String	
RemovePiece (public) <<virtual>>		
Parameters		Used for removing a piece fron
Return values	PieceToReturn : Piece	The method makes a temporar attribute PieceInSquare in a lo then sets the attribute to null to It then returns the variable Piec
SetPiece (public) <<virtual>>		
Parameters	P : Piece	Assigns the P parameter to the PieceInSquare.
Return values	n/a	

Class: Kotla (inherits from Square)

Identifier / Data		Des…on
<<constructor>>		
Parameters	P : Player S : …	Initialises the following parent a • BelongsTo from parame • Symbol from parameter S
Return va…	n/a	
GetPointsForOccupancy (public) <<override>>		
Parameters	CurrentPlayer : Player	Overrides the GetPointsForO base class to return the score p occupied. The method checks first to see square. If there is not, the meth If there is a piece in the Kotla s to see if the Kotla square belon CurrentPlayer passed in as a p the piece in the Kotla is either a also ow…d by the CurrentPla p…s. Kotla square belongs t…e is no Mirza or standard pi zero points. If the Kotla square belongs to th piece in it is either a Mirza or a CurrentPlayer, the method ret returns zero points.
Return values	Integer	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Class: MoveOption

Identifier / Data		Description
<<constructor>>		
Parameters	N : String	Initialises the following p • ame from parame • PossibleMoves to
Return values	n/a	
AddToPossibleMoves (public)		
Parameters	M : M	Adds the M parameter to list.
Return va		
CheckIfTh sAMoveToSquare (public)		
Parameters	StartSquareReference : Int FinishSquareReference : Int	Used to check if the star by the player are valid st MoveOption. The method initialises fo and StartColumn togeth FinishColumn. The met StartRow and MOD to s the StartSquareReferen same techniques to split FinishColumn from the parameter. Th ethod then iterates ossibleMoves list chec StartColumn and Finis combination represent a possible positions a piec
Return values	Boolean	
GetName)		
Parameter	n/a	Returns the value of the
Return values	Name : String	

Class: Move

Identifier / Data		Description
<<constructor>>		
Parameters	R : Int C : Int	Initialises the following protected • RowCh ge from paramet • C m hange from para
Return values	n/a	
GetColumnChange (public)		
Parameters	n/a	Returns the value of the protecte
Return val	nge : Int	
GetRowC (public)		
Parameters	n/a	Returns the value of the protecte
Return values	ColumnChange : Int	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Class: MoveOptionQueue

This class does not have a specific constructor and therefore uses the default constru

Identifier / Data		Description
<<constructor>>		
Parameters	n/a	Initialises the Queue p MoveOption list.
Return values	n/a	
Add (public)		
Parameters	NewMoveOption : MoveOption	Adds the NewMoveOp Queue list.
Return va	n/a	
GetMoveOptionInPosition (public)		
Parameters	Pos : Int	Returns the MoveOpti Queue list.
Return values	MoveOption	
GetQueueAsString (public)		
Parameters	n/a	Initialises a local empty QueueAsString and a which it assigns 1. The method then iterat concatenating the Cou name of each Move in GetName() method), in each loop. The method then return variable.
Return values	QueueAsString : String	
MoveItemToBack (public)		
Parameter	Position : Int	Used for moving a Mov Queue list. The method makes a te MoveOption at the ind The method then uses on the Queue list to re index Position. It then appends the ten MoveOption back into effect of placing it at the
Return values	n/a	
Replace (public)		
Parameters	Position : Int NewMoveOption : MoveOption	Assigns the NewMove Queue list at the index parameter.
Return values	n/a	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Class: Player

Identifier / Data		Description
<<constructor>>		
Parameters	N : String D : Int	Initialises the following p • core to 100 • Name from parame • Direction from para
Return values	n/a	
AddToMoveOptionQueue (public)		
Parameters	veOption : MoveOption	Adds the NewMoveOpti Queue attribute.
Return values	n/a	
ChangeScore (public)		
Parameters	Amount : Int	Increments the protected Amount parameter.
Return values	n/a	
CheckPlayerMove (public)		
Parameters	Pos : Int StartSquareReference : Int FinishSquareReference : Int	Used to check if a move using the CheckIfThere The method creates a te move selected from the parameter. The method then passes and FinishSquareRefer CheckIfThereIsAMoveT the references represent selected move option.
Return values	Boolean	
GetDirection (public)		
Parameters	n/a	Returns the value of the
Return values	Direction : Int	
GetName (public)		
Parameters	n/a	Returns the value of the
Return values	Name : String	
GetPlayerStateAsString (public)		
Parameters	n/a	Used to expose the Get the MoveOptionQueue through the player. The method returns a co attribute and the player string using the GetQue
Return values	String	
GetScore (public)		
Parameters	n/a	Returns the value of the
Return values	Score : Int	

INSPECTION COPY
COPYRIGHT PROTECTED

SameAs (public)		
Parameters	APlayer : Player	Used to check if the AP as this player object. The method first checks object has been passed fals if it is null. If not, the method comp parameter with the nam match, the method retu they are the same play
Return values	Boolean	

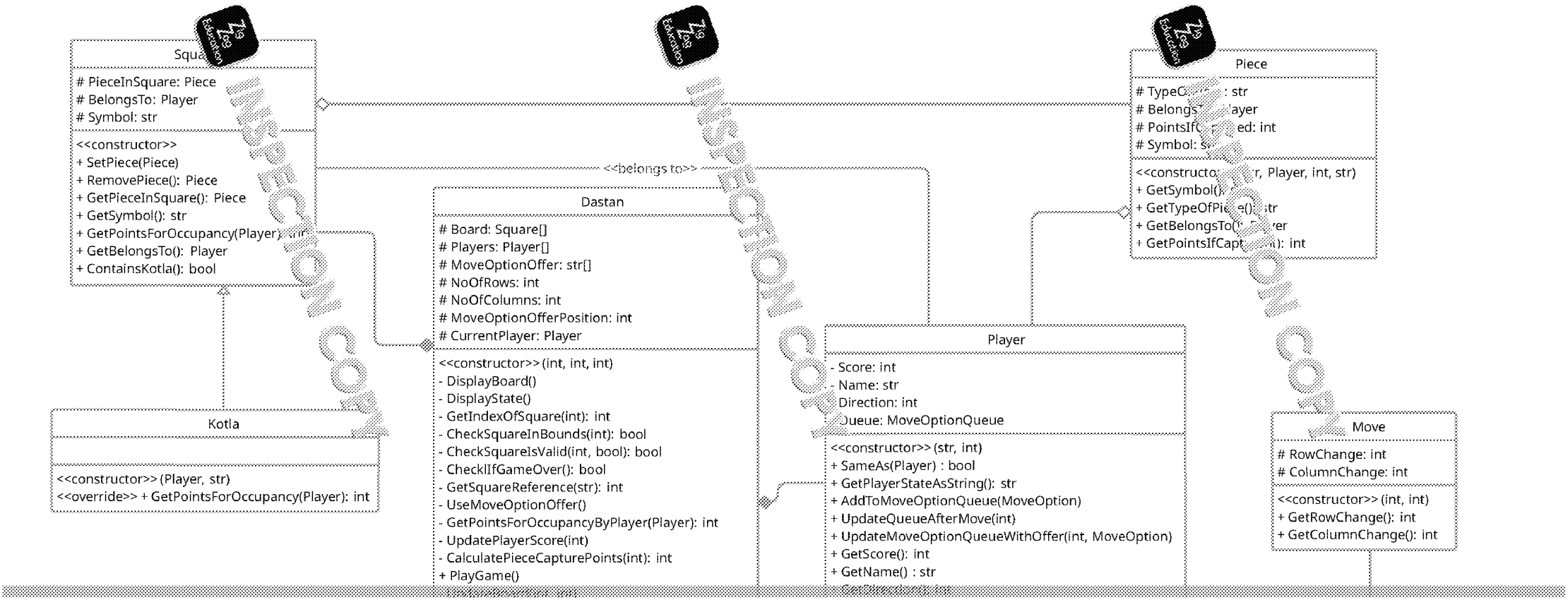
UpdateM licQueueWithOffer (public)		
Parameters	Position : Int NewMoveOption : MoveOption	Used to expose the Re MoveOptionQueue cla the player. The method calls the R queue, passing the Pos parameters. This will re index of Position with t parameter.
Return values	n/a	

UpdateQueueAfterMove (public)		
Parameters	Position : Int	Used to expose the Mo MoveOptionQueue cla the player. The method calls the M the player queue passi minus one to make it ze move option at that inde moved to the back of th
Return values	n/a	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Dastan

UML Class Diagram



Zig Zag Education

COPYRIGHT PROTECTED

INSPECTION COPY

Dastan

Exam-style Questions

These questions refer to the **Preliminary Material** and the **Sk**
but **do not** require any additional programmin

TOTAL MARKS: 60

1 This question refers to the PlayGame method in the Dastan class.

The method contains a nested loop with multiple while loops inside th

(a) State the time complexity of this loop.

.....

(b) Explain the efficiency of this time complexity and how well it scale

.....

.....

.....

.....

.....

2 This question refers to the entire pre-release code.

Throughout the code there are many literals such as 'mirza', 'jazair', 'n
others.

(a) Describe one problem that could occur due to this.

.....

.....

.....

(b) Describe one poss[illegible] solution to this problem.

.....

.....

.....

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

3 This question refers to the private method GetPointsForOccupancyBy

Explain precisely how polymorphism is used when calculating the Scor

.....

.....

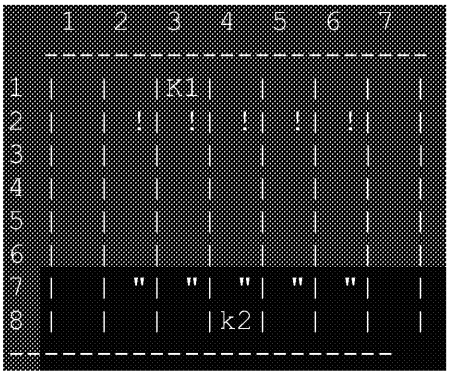
.....

.....

.....

4 This question refers to the Main method that is executed at the start o

When ThisGame is instantiated, currently the arguments 6, 6, 4 are pa



(a) Assume the arguments 8, 7, 6, 4 are passed to Dastan instead.

Explain why player Kotla and Mirza would appear in column opposite the two's as per the image above.

.....

.....

.....

(b) Describe how the code for the CreateBoard method of the Dastan so that where there are an odd number of columns, then the Kotla central column but when there are an even number it will remain a

.....

.....

.....

.....

.....

.....

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

5 This game refers to the private methods CreateRyottMoveOption, CreateFaujdarMoveOption,CreateJazairMoveOption, CreateCuirassi CreateChowkidarMoveOption.

Currently the methods take a Direction parameter which changes bet to whose turn it is. Across the methods there is a lot of repeated use o parameter which always gets multiplied by any non-zero parameter to

Without suggesting any specific code, de.. e alternative logic that c the Direction parameter by modif.. he AddToMoveOptionQueue UpdateMoveOptionQue.. t.. er methods of the Player class.

.....

.....

.....

.....

.....

6 This question refers to the MoveOptionQueue class.

The game uses a queue data structure rather than a stack.

(a) Explain why a queue is a more suitable data structure than a stac

.....

.....

.....

(b) Currently this class uses a list to store the queue data structure; e modified to use an array to implement a circular queue with five e

You should not write any actual code for this question but refer to may be required and create any algorithms using structured/desc Alternatively, you may produce an annotated diagram.

.....

.....

.....

.....

.....

.....

.....

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

7 This question refers to the method GetIndexOfSquare in the Dastan

Explain how the private method GetIndexOfSquare works.

.....

.....

.....

.....

.....

8 The board is currently represented as a one-dimensional array, but th
alternative representations.

(a) Explain how the board could be represented as a two-dimensiona

.....

.....

.....

(b) State one reason why an array is more appropriate to store the bo

.....

.....

9 It woul possible to create a save game file for Dastan. At the start
metadata.

Explain the purpose of metadata and give one example of metadata th
Dastan.

.....

.....

.....

.....

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

10 This question refers to the CreateMoveOptions, CreateMoveOption, CreateChowkidarMoveOption, CreateRyottMoveOption, CreateFaujd CreateJazairMoveOption and CreateCuirassierMoveOption methods the MoveOption class.

(a) Currently the MoveOption class holds the details for whichever n generated/populated by one of the CreateChowkidarMoveOptio CreateRyottMoveOption, CreateFaujdarMoveOption, CreateJ CreateCuirassierMoveOption methods in the Dastan class.

Explain why this is NOT polymorphism.

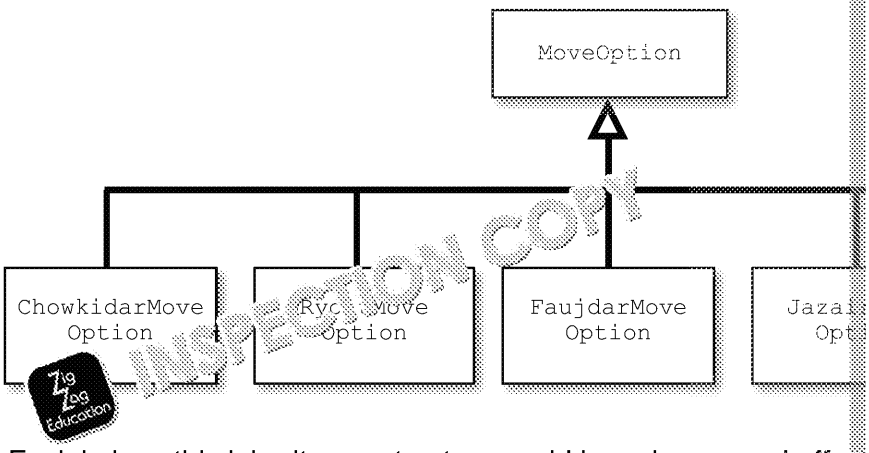
.....

.....

.....

.....

(b) An alternative would have been to create and use an inheritance s following:



Explain how this inheritance structure could have been used effec

.....

.....

.....

11 This question refers to the Kotla class.

(a) The constructor includes a call using sup e plain the purpose

.....

.....

.....

(b) The method GetPointsForOccupancy has a different implement with the same name in the parent class. State the name for this C

.....

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

11 (c) Explain what the OOP technique *overloading* is used for.

.....

.....

.....

.....

.....

12 The MoveOptionQueue class implements a normal queue, which is a data structure.

Explain the different between a normal queue and a priority queue.

.....

.....

.....

.....

.....

.....

13 This question is about the constructor of the Piece class and the SetP Squares.

Both methods take a parameter *P* which is unclear. Explain why variab meaningful names.

.....

.....

.....

14 This question is about access levels for attributes and methods and re

(a) The Piece class has four protected attributes, what does the word context?

.....

.....

.....

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

14 (b) The Piece class has four public methods; what does the word 'pu

.....

.....

(c) There is one final level of access for attributes and methods which mean?

.....

.....

(d) W[illegible]t important to have access modifiers such as private, prote methods and attributes in OOP?

.....

.....

.....

.....

.....

15 This question refers to the CheckSquare[illegible]ds method of the Da

(a) This method uses integ[illegible]isi[illegible], explain the difference between floating point div[illegible]

.....

.....

.....

(b) This method returns a Boolean value. Describe the meaning of B

.....

.....

END OF QUESTIONS

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Dastan

Exam-style Questions

These q[illegible] refer to the **Preliminary Material** and the **Sk**
but **do not** require any additional programmin

TOTAL MARKS: 60

1 This question refers to the PlayGame method in the Dastan class.

The method contains a nested loop with multiple while loops inside th

(a) State the time complexity of this loop.

(b) Explain the efficiency of this time complexity and how well it scale

2 This question refers to the entire pre-release code.

Throughout the code there are many literals such as 'mirza', 'jazair', 'n
others.

(a) Describe one problem that could occur due to this.

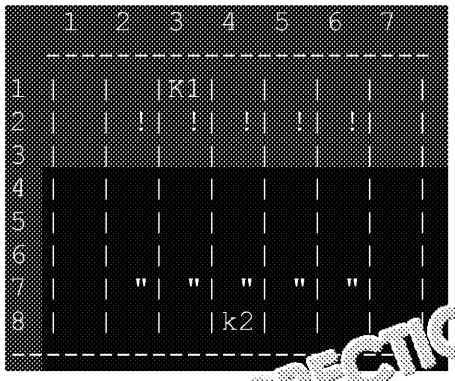
(b) Describe one possible solution to this problem.

3 This c[illegible]n refers to the private method GetPointsForOccupancyBy

Explain precisely how polymorphism is used when calculating the Score

4 This question refers to the Main method that is executed at the start o

When ThisGame is instantiated, currently the arguments 6, 6, 4 are p



(a) As[illegible]e t[illegible]guments 8, 7, 5 were passed to Dastan instead.

Ex[illegible] why player one's Kotla and Mirza would appear in colum
opposite player two's as per the image above.

(b) Describe how the code for the CreateBoard method of the Dastan
so that where there are an odd number of columns, then the Kotla
central column but when there are an even number it will remain a

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

5 This game refers to the private methods CreateRyottMoveOption, CreateFaujdarMoveOption,CreateJazairMoveOption, CreateCuirassi CreateChowkidarMoveOption.

Currently the methods take a Direction parameter which changes bet to whose turn it is. Across the methods there is a lot of repeated use o parameter which always gets multiplied by any non-zero parameter to

Without suggesting any specific code, de[illegible]e alternative logic that c the Direction parameter by modif[illegible]he AddToMoveOptionQueue UpdateMoveOptionQue[illegible]er methods of the Player class.

6 This q[illegible]n refers to the MoveOptionQueue class.

The game uses a queue data structure rather than a stack.

(a) Explain why a queue is a more suitable data structure than a stac

(b) Currently this class uses a list to store the queue data structure; e modified to use an array to implement a circular queue with five e

You should not write any actual code for this question but refer to may be required and create any algorithms using structured/desc Alternatively, you may produce an annotated diagram.

7 This question refers to the method GetIndexOfSquare in the Dastan

Explain how the private method GetIndexOfSqu[illegible] works.

8 The board is currently rep[illegible]as a one-dimensional array, but th alternative repres[illegible]

(a) E[illegible]h[illegible] the board could be represented as a two-dimensiona

(b) Sta[illegible] one reason why an array is more appropriate to store the bo

9 It would be possible to create a save game file for Dastan. At the start metadata.

Explain the purpose of metadata and give one example of metadata t Dastan.

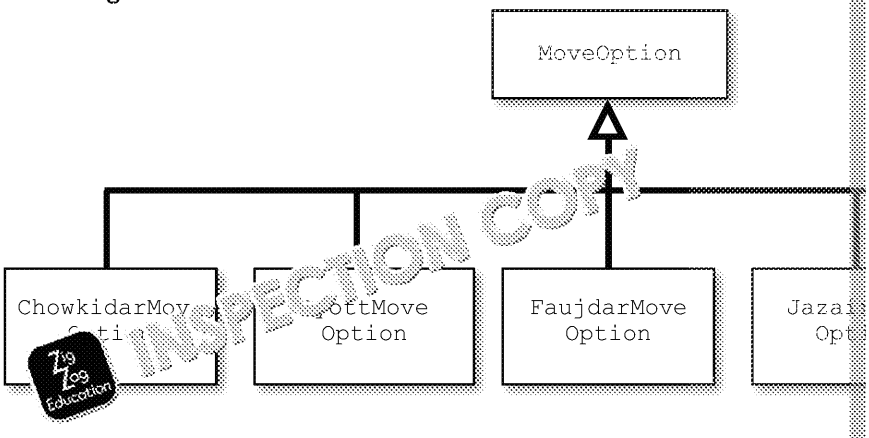
10 This question refers to the CreateMoveOptions, CreateMoveOption, CreateChowkidarMoveOption, CreateRyottMoveOption, CreateFaujd CreateJazairMoveOption and CreateCuirassi[illegible]veOption methods the MoveOption class.

(a) Currently the MoveOp[illegible]s holds the details for whichever n generated/po[illegible]one of the CreateChowkidarMoveOptio Cr[illegible]RyottMoveOption, CreateFaujdarMoveOption, CreateJ C[illegible]uirassierMoveOption methods in the Dastan class.

Explain why this is NOT polymorphism.

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

10 **(b)** An alternative would have been to create and use an inheritance s
following:



Explain how this inheritance structure could have been used effec

11 This question refers to the Kotla class.

(a) The constructor includes a call using super(); explain the purpose

(b) The method GetPointsForOccupancy has a different implement
with the same name in the parent class. State the name for this C

(c) Explain what the OOP technique *overloading* is used for.

12 The MoveOptionQueue class implements a normal queue, which is a
data structure.

Explain the different between a n queue and a priority queue.

13 This qu on the constructor of the Piece class and the SetP
Squa s.

Both methods take a parameter *P* which is unclear. Explain why varia
meaningful names.

14 This question is about access levels for attributes and methods and re

(a) The Piece class has four protected attributes; what does the word
context?

(b) The Piece class has four public methods; what does the word 'pu

(c) There is one final level of access for attributes and methods which
mean?

(d) Why is it important to have access mo ier such as private, prote
methods and attributes in OOP

15 This qu the CheckSquareInBounds method of the Da

(a) Th hod uses integer division; explain the difference between
floating point division.

(b) This method returns a Boolean value. Describe the meaning of E

END OF QUESTIONS

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Dastan

Programming Tasks

These questions requ[illegible] load the **Skeleton Program** and to make

Note that a[illegible]native or additional code changes that you deemed appropriate [illegible]suring that it is clear where in the Skeleton Program those chang[illegible]

Important: Throughout this document and the Python code, methods are referred public. In this document, method names are written **without** leading underscores, method names are written **with** leading underscores; a private method appears wi[illegible] start and a protected method with a single underscore.

Task 1

This question refers to the Dastan class.

Introduce new functionality at the point at which both players are instantiat[illegible] custom names set by the users. Ensure that players cannot both have the [illegible] replace the two lines in the constructor that cu[illegible]reate the players wi[illegible] method, CreateCustomPlayers.

What you need to do

Task 1

Create a new method CreateCustomPlayers in the Dastan class. Allow th[illegible] names for each player. Include checks in your code to ensure that two play[illegible] custom name.

Allow the first player to enter any name they like, then repeatedly ask the u[illegible] name until they are both different.

Task 2

Test that the changes you have made work:

- run the skeleton program.
- enter 'Tom' as the first player name and then enter 'Tom' as the se[illegible] prompted, enter 'Tom' again and then at the next prompt, enter 'Vic[illegible]
- show the game [illegible] of the custom names to address the play[illegible]

Evidenc[illegible] you need to provide:

- PROGRAM SOURCE CODE showing creation of a new CreateCus[illegible] Dastan class
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Task 2

This question refers to the CreateMoveOptionOffer, CreateMoveOption
methods and creation of a new method CreateFarisMoveOption in the D

Develop a new move option called a 'Faris' (Knight). The Faris move optio
chess – either two squares forward/backwards ar[illegible] square left/right o
left/right and one square forward/backwa[illegible]. You should demonstrate the
parameter.

What you [illegible] to

Task 1

i) Add new functionality into the CreateMoveOptionOffer & CreateMoveOption methods to perform a Faris move.

ii) Modify the CreateMoveOptions method to add the Faris after the Ryott for both players.

iii) Create a new method CreateFarisMoveOption which adds moves using the pattern shown, to the NewMoveOption object.

Task 2

Test that the changes you h[illegible] m[illegible] work:

- run[illegible] program.
- play[illegible] urns, showing both players making legal Faris moves.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the Create
CreateMoveOption and CreateMoveOptions methods
- PROGRAM SOURCE CODE showing a new method CreateFarisM
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Task 3

Develop a new move option called a 'Sarukh' (Rocket). The Sarukh move c
rocket shape. You should demonstrate the use of the Direction paramete

What you need to do

Task 1

i) Add new fu[illegible]ty into the CreateMoveOptionOffer, Cre[illegible]veOption and CreateMoveOptions methods to pe[illegible]rm a Sarukh move.

ii) Modify the CreateMoveOptions method to add the Sarukh after the Ryott for both players.

iii) Create a new method CreateSarukhMoveOption which adds moves using the pattern below, to the new MoveOption object. The pattern is shown from the viewpoint of player two. For player one, the layout is inverted.

Task 2

Test that the changes you have made work:

- run the skeleton pro[illegible]m
- play two tu[illegible]s showing both players making legal Sarukh moves.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the Create
  CreateMoveOption and CreateMoveOptions methods
- PROGRAM SOURCE CODE showing a new method CreateSarukh
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Task 4

This question refers to the PlayGame method in the Dastan class and cre
AwardWafr in the Dastan class, GetWafrAwarded and SetWafrAwarde
attribute WafrAwarded in the Player class.

Create a 'Wafr' (abundance) award which can be awarded to either player o
a 25% chance of being awarded to a pla[illegible] on their turn. On receipt of the
option of ANY move from their move queue rather than just being able to s
The 'Wafr' award removes the move cost for the move the player selects fo

Note: If th[illegible]er makes an invalid move then they 'lose' their Wafr and g
player should not be able to 'take the offer' if a Wafr is awarded.

What you need to do

Task 1

i) Create a new method in the Dastan class called AwardWafr. This
chance of returning true.

ii) Add a new private attribute to the Player class called WafrAwarde
mutator (getter/setter) methods for this attribute.

Task 2

Update the PlayGame method in the Dastan cla[illegible] call the new Award
hasn't already been awarded a Wafr, [illegible]ut a message saying 'You hav
can select any move from your queue for free this turn.' Adjust the input ra
in the queue to be selected. Ensure that there is no score adjustment for p
value of th[illegible]ou[illegible] to ensure that they cannot receive another Wafr.

Task 3

Test that the changes you have made work:

- run the skeleton program.
- play the game to show a player being awarded a Wafr.
- play a move option from position 4 or 5 in the move option queue.
- show the updated board and correctly modified score.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGam
class, creation of a new meth[illegible]dWafr in the Dastan class
- PROGRAM SOURCE CODE showing changes made to the Player c
meth[illegible]etWafrAwarded, SetWafrAwarded together with one new
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 5

This question refers to the PlayGame method in the Dastan class and the GetJustQueue in the Player class.

Introduce a new option 8 to the main game playing menu. On selecting this their opponent's queue to spy what move options their opponent might be an opponent's queue, however, carries a cost of 5 points from the player's opponent's queue, the player's turn should continue as normal.

What you need to do

Task 1

Create a new method in the Player class called GetJustQueue which uses method to return a string version of just the player's queue.

Task 2

Modify the PlayGame method to introduce new functionality which adds a game playing menu. If the user selects this option, display the move option player.

(**Hint:** You can check the current player using the SameAs method and th Subtract 5 from the current player score and display the game state again their turn as normal.

Task 3

Test that the changes you have made work:

- run the skeleton program.
- show player one selecting option 8 from the main game menu.
- show the opponent queue being displayed clearly on the screen an reducing by 5 points.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa class
- PROGRAM SOURCE CODE showing new method GetJustQueue i
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Task 6

This question refers to the PlayGame method together with the modificatio
UseMoveOptionOffer methods and creation of a new method GetValidIn

Currently the game has a number of areas where it does not handle errors
error handling into the PlayGame, GetSquareRe[illegible]nce and UserMoveO
prevent unhandled exceptions from occurring if the user inputs data in an i
user to re-enter their input, until it is valid.

Note: There is no need to check that the square contains a player piece o
player sho[illegible] have a wasted turn if the move is invalid, the purpose of t
from crashing.

What you need to do

Task 1

Create a new private method called GetValidInt in the Dastan class which
valid integer. If the input is invalid, allow the user to keep trying again with

Task 2

Modify the GetSquareReference method to use the new GetValidInt meth
input. Add an error message if the user enters an invalid square.

Task 3

Modify the UseMoveOption[illegible] method to use the new GetValidInt met
input and test to confirm that the user input is within the correct range.

Task 4

Test that the changes you have made work:

- run the skeleton program.
- from the main game playing menu, enter 'help' as your choice and
message. Then choose move 1.
- For player one, enter a square of 19 and show the error message.
followed by 32 to make the move.
- For player two, select option 9 to take the offer move and choose p
message.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the GetSqu
- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE showing changes made to the UseMov
- PROGRAM SOURCE CODE showing the creation of new GetValidI
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 7

This question refers to the PlayGame and UseMoveOptionOffer methods creation of a new attribute ChoiceOptionsLeft along with accessor and m DecreaseChoiceOptionsLeft and GetChoiceOptionsLeft in the Player c

Currently a player can repeatedly select option 9 from the main game play with new move options. Introduce a limit so that a player can only 'accept t menu three times in a game. Each time a player accepts the offer, advise t they have left and remove the menu for that player once they have used it

What you need to do

Task 1

Modify the Player class to introduce a new private attribute called Choice

i) Initialise ChoiceOptionsLeft to 3.

ii) Create a new accessor method called GetChoiceOptionsLeft whi attribute ChoiceOptionsLeft.

iii) Create a new mutator method called DecreaseChoiceOptionsLef ChoiceOptionsLeft attribute and prints out how many options you

Task 2

Modify the PlayGame method to test the number of options the player has three during the game.

i) Modify the Player class method so that if the player has used up all t will no longer be available to the player.

ii) Modify the UseMoveOptionOffer method so that when a move op the number of options available to them decreases by one.

Task 3

Test that the changes you have made work:

- run the skeleton program.
- select four sequential option moves from the move option list addin the player one queue.
- show the removal of option 9 from the main game playing menu an the player attempts to select option 9.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGam
- PROGRAM SOURCE CODE showing changes made to the UseMov Dastan class
- PROGRAM SOURCE CODE showing changes made to the Player c
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
COPYRIGHT PROTECTED

Task 8

This question refers to the PlayGame method in the Dastan class and creat
ResetQueueBack in the MoveOptionQueue class and ResetQueueBack

Introduce a new option that allows a player to undo their last move (after t
and before the next player makes their move), u[illegible] any score gained o
returning the game to its previous state. Undoing a move costs a player 5
a player can then make an alte[illegible]ove.

What you [illegible] to do

Task 1

Add the functionality to reset the queue if a move is undone.

i) Create a new method in the MoveOptionQueue class called Rese
should move the last element of the queue back to the original pos
method should take one parameter, Position, which is the place to
queue will be restored.

ii) Create a new method in the Player class called ResetQueueBack
should call the newly created ResetQueueBack method on the Qu
class. The method should take one parameter, Position, which is t
made from the menu.

Task 2

Modify the PlayGame [illegible] introduce the new functionality.

i) If a [illegible] is legal, store the player score prior to the move.
ii) After [illegible]playing the board as a result of the move, give the player t
iii) If they choose to undo then: return the player score to the stored p
points and restore the board and the player's queue back to their p

Task 3

Test that the changes you have made work:

- run the skeleton program.
- show player one attempt a 'Chowkidar' move and then undo the m
- show the game board after the undo and the score set correctly an
a new move.

Evidence that you need to pro[illegible]

- PROGRAM SOUR[illegible] showing changes made to the PlayGam
- PROG[illegible]S[illegible]URCE CODE showing the creation of new methods R
Move[illegible]Queue class
- PROGRAM SOURCE CODE showing the creation of the new method
ResetQueueBackAfterUndo in the Player class
- SCREEN CAPTURE(S) showing the required test

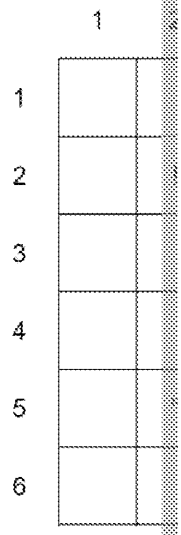
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Task 9

This question refers to the PlayGame method together with the modificati
and CreateMoveOption methods and creation of two new methods, Crea
CalculateSahmMove, in the Dastan class – plus a new method, ChoiceI

It also refers to a new attribute SahmUsed in the Player class along with
GetSahmStatus and SetSahmUsed, which operate as the accessor and
methods for the newly created SahmUsed attribute.

Implement a new 'Sahm' move option (arrow). The Sahm can only be fired
and is fired instead of a piece moving. A Sahm can be fired by any piece.
line forwards from the player destroying any opponent piece(s) in its way except a Kotla, which is strong enough to withstand an attack and protect any piece inside it. The Sahm is only made available to a player through the MoveOptionOffer method (they can choose to add it to their moves by using option 9 from the main menu at the start of the turn if a Sahm is offered to them). A Sahm will not show up normally in the MoveOptionQueue.



The image on the right shows the player 2 piece in square 54 firing the Sahm. The Sahm will fire forwards, destroying the player 1 pieces in squares 34 and 24.

What you need to do

Task 1

Add new functionality into the CreateMoveOptionOffer and CreateMove
new private CreateSahmMoveOption method to perform a Sahm move.

i) Modify the CreateMoveOptionOffer method to offer the new 'Sahm

ii) Create the new private CreateSahmMoveOption method to allow
piece fires the Sahm and add only one possible new move Move(0
Note: The move should not actually move the piece anywhere, i.e.

iii) Modify the CreateMoveOption method to handle Sahm.

Task 2

Modify the Player class to allow the user to use the Sahm only once.

i) Add a new SahmUsed attribute to the Player class which is initialis

ii) Create two new methods, GetSahmStatus and SetSahmUsed, wh
and to (getter/setter) methods for the newly created SahmUs

iii) Create a method ChoiceIsSahm method which takes a parameter
chosen is a Sahm move, whereupon it returns True.

(TASK CONTINUES ON THE NEXT PAGE)

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 3

Modify the PlayGame method to test to see if the player has selected a Sa
MoveOptionOffer menu and if it has already been used. If the selected fir
should destroy any opponent pieces in a straight line from the firing piece,
player should collect any points from multiple pieces destroyed by the Sah

i) Modify PlayGame to call the new method ChoiceIsSahm and only

ii) Create a new private method in the Dastan class called CalculateS
calculate the point of the Sahm move and destroy the pieces that a
Kotla)

iii) Modify PlayGame to so that is calls the new method CalculateSah
the Sahm move and destroys the relevant pieces. It should also ca
for the current player.

Task 4

Test that the changes you have made work:

- run the skeleton program.
- select a Chowkidar move for player one (option 2) and choose squa
square 33 as the 'to' to diagonally move one piece in front of anoth
Kotla column.
- select 9 from menu for player two to accept the offer. Choose 1 to
choose option 1 to select the Sahm move. Choose the piece on sq
show the updated board with both player one pieces removed from
by player two, but [illegible] Mirza which is safely inside the Kotla.
- show the correct adjustment of player two's score.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE showing changes made to the CreateM
CreateMoveOption methods
- PROGRAM SOURCE CODE showing the creation of new CreateSa
ChoiceIsSahm and CalculateSahmMove methods
- PROGRAM SOURCE CODE showing changes made to the Player c
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 10

This question refers to the PlayGame method in the Dastan class.

Introduce a new option 7 to the main game playing menu. On selecting this one of their own pieces to destroy and replace with a second Kotla. A new the square in which the piece was sacrificed. A player can only replace one Replacing a piece with a Kotla should use up a player turn and they should turn.

What you need to do

Task 1

Modify the PlayGame method in the Dastan class to introduce a new optio playing menu. Allow the player to select a piece which they would like to re validation to ensure that the user can only select one of their pieces and it confirmation, replace the piece with a second Kotla assigned to the correct

Task 2

Test that the changes you have made work:

- run the skeleton program.
- select option 7 for player one from the main game menu.
- show the user selecting 52 as an invalid square for the new Kotla.
- show the Kotla being placed correctly in square 22, a valid square,

Evidence you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Task 11

This question refers to the PlayGame method together with a new method in the Dastan class, additional new methods ReverseQueue, SwapFirstA MoveItemToFront in the MoveOptionQueue class together with new me SwapQueue, GetMoveOptionQueue, ReversePlay Queue, SwapFirst MoveItemToFront in the Player class.

Introduce a new option 6 to the main game playing menu. On selecting thi sub options for making changes to their move queue using the following m

Options
a) Reverse the current player queue
b) Swap the current player queue with the opponent queue
c) Swap the first and last elements in the current player queue
d) Move one of the move options to the front of the current player queue
e) Nothing (make normal move)

Note: Options (a) – (d) cost 3 points, but the player can choose (e) for free

Note: This does not count as the player's turn and the player should still be

What you need to do

Task 1

Modify the main program to introduce the new menu option.

i) Modify the PlayGame method to add option 6 to the move options

ii) Create a new private method in the Dastan class called ModifyQu player the above menu. Include validation to ensure that the user c choices from the menu.

iii) Adjust the score by 3 if options (a) – (d) are chosen but not if option

Task 2

Modify the MoveOptionQueue class to add the required methods.

i) Create new method ReverseQueue to allow the current player's q

ii) Create new method SwapFirstA to swap the first and last e queue.

iii) Create new method MoveItemToFront to move the item from the the front for the current player. There is no need to validate the i the position from.

(TASK CONTINUES ON THE NEXT PAGE)

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 3

Modify the Player class to create the required methods.

i) Create new methods ReverseQueue, SwapFirstAndLast, MoveIt class to expose the new MoveQueueOptions choices/methods to

ii) Create new method ReplaceQueue to allow the current player's qu queue passed in as a parameter. Note that it should return the cur

Task 4

Test that the changes you have made work:

- run the skeleton program.
- show player one selecting option 6 from the main game menu.
- show the player selecting each one of the queue options in turn an screen as a result of the change.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE for the new ModifyQueueOptions met
- PROGRAM SOURCE CODE showing changes made to the MoveOp
- PROGRAM SOURCE CODE showing changes made to the Player c
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Task 12

This question refers to the creation of a new protected attribute NoOfPiec
PlayGame method and creation of two new methods CheckReincarnatio
the Dastan class.

Introduce a new feature whereby if a player manages t get one of their pi
row, they are given a new piece to place on an unoccupied space on thei
player cannot reincarnate pieces that are not dead so they should not be a
they started with.

What you need t

Task 1

Create a new private method in the Dastan class called CountNormalPie
of pieces that the current player has excluding the Mirza.

Task 2

i) Modify the constructor in the Dastan class to store the number of p
protected attribute called NoOfPieces.

ii) Modify the PlayGame method in the Dastan class to call a new pri
CheckReincarnation after the move is legal.

Task 3

Create a new private method CheckReincarnation in the Dastan class. T
which is the FinishSquareReference for the current player's move. If the
opponent's back row (e.g. row 6 for player one) and the player has fewer p
then allow them to reincarnate a piece on their back row in an empty squa
the square is empty and allow them to reincarnate it is not.

Task 4

Test that the changes you have made work:

- add the following four lines of code to the START of the private me
Dastan class (*be certain to remove this afterwards!*):

```
NoOfPieces = 2
self._Board[self.__GetIndexOfSquare(51)].SetPiece(Piece("piece"
self._Board[self.__GetIndexOfSquare(21)].SetPiece(Piece("piece"
self._Board[self.__GetIndexOfSquare(54)].SetPiece(Piece("piece"
```

- run the skeleton program.
- select a Ryott move for player one, enter a start square of 51 and a
- show player one attempting to reincarnate a piece in column 3 and
saying that the square must be empty.
- show player one attempting to reincarnate a piece in column 4 and
appropriately.
- select a Ryott move for player two, enter a start square of 21 and a
- show player two not receiving a reincarnation message.
- Change back the CreateNewBoard method by removing the additiona

Evidence you need to provide:

- PROGRAM SOURCE CODE showing the new CountNormalPieces
- PROGRAM SOURCE CODE showing the new CheckReincarnatio
- PROGRAM SOURCE CODE showing the other code changes to the
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 13

This question refers to the PlayGame method together with modification o
the Dastan class. Additionally it involves the creation of a new Taziz class

Create a new type of game square, the Taziz (advantage castle, similar to
the middle of the playing board (or slightly closer to player two if there are
Either player can occupy the Taziz with any of their pieces. If a player can
by both players (entering the taziz is considered a player's first turn), then
have zero cost. This gives a player a zero cost move, but risks sitting in the
to get it. If a player sits there for longer then they continue to get zero co

What you need to do

Task 1

Create a new Class Taziz which should inherit from the Square class.

i) Add a new protected attribute CampedTurns and initialise it to 0.

ii) Override the SetPiece and RemovePiece methods from the Squa
adjust the Taziz symbol to an upper case 'A' if player one owns the
player two owns the Taziz (you may assume that the player with a
the top – player one). When a player piece leaves the Taziz , owne
set to null and the symbol set to a lower case 'x'.

iii) Create a new method GetCampedTwoTurns. Each time the Taziz
CampedTurns should be reset back to zero. The GetCampedTwo
the number of turns using the CampedTurns attribute and return t

iv) Create a new method CheckCamp that checks if the same player
inc[illegible]s the CampedTurns attribute if they are.

Task 2

Modify the CreateBoard method in the Dastan class to place a Taziz on th
middle of the board with a lower case 'x' symbol when the board is first cre

NOTE: *The Taziz should be correctly placed on the board even if the size
should take account of the number of columns and rows.*

In the case where there are an even number of rows, the Taziz should be
also if there are an even number of columns then it should be slightly close
starting board this will place it on square 43, but it should work for any size

The initial Taziz does not belong to either player.

Task 3

Modify the [illegible]ame method so that if a move is legal the game should t
been camped in for two full turns and, if so, give the selected move to the p

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 4

Test that the changes you have made work:

- run the skeleton program.
- use a Cuirassier move option 3 to move a player one piece into the
- play the game until both players have had two turns – leaving the p without attacking it using player two.
- after both players have had two turns, show a move option by playe

Evidence you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE showing changes made to the CreateB
- PROGRAM SOURCE CODE showing the new GetCampedTwoTur Square class
- PROGRAM SOURCE CODE showing the new Taziz class
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Task 14

This question refers to the PlayGame method together with creation of a n
WeatherEventOccurs method in the Dastan class. Additionally it involves
WeatherEvent with the methods CountDownComplete, SetWeatherLoc
GetWeatherLocation.

The Weather Event has a 50% chance of appearing on any turn and can a
space on the board. On appearance on the board, both players are given a
Weather Event will destroy EVERY piece on the same column as the Wea
After two turns by each player, the Weather Event strikes and any piece fr
column is removed, including the Kotla.

***NOTE:** A Weather Event can only occur if a Weather Event is not already*

What you need to do

Task 1

Create a new class WeatherEvent which should include new methods Co
SetWeatherLocation and GetWeatherLocation. On instantiation, the We
countdown to count the number of game turns before the event occurs. Co
test to see if the countdown has expired. The SetWeatherLocation and G
should set and get the location of the Weather Event on the board. Suitab
out each turn to indicate how long until the Weather Event will occur.

Task 2

Create a new method called WeatherEventOccurs in the Dastan class w
creating a Weather Event and setting it to a random empty square on the boar
occurred, let the player know.

Task 3

Modify the PlayGame method in the Dastan class to test to see if a Weath
so if the Weather Event countdown has expired. If it has, use the Weather
piece (from either player) from the same column as the Weather Event, in
are awarded for this event.

Task 4

Test that the changes you have made work:

- run the skeleton program.
- when a weather event occurs, move player pieces to be on the sam
  event over the next two turns.
- show the board during the countdown to the Weather Event and aft
  showing the pieces from both players removed from the board.

Evidence you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE showing the new WeatherEventOccu
- PROGRAM SOURCE CODE showing the new WeatherEvent class
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 15

This question refers to the PlayGame method together with modification o
and CreatePieces methods and creation of three new private methods, C
PlaceBarrier and CheckManhattanDistance in the Dastan class. Additio
new public method ContainsBarrier in the Square cl and the creation
inherits from Square.

Create a new game piece calle . On creation of the board each p
would like to place th on the board. The Barrier is 3 squares wid
the board p tion occupied by a normal piece or an opponent's Ba
be moved, ied or jumped over by either player.

Some moves, however, do not move in a straight line, for example the Jaz
the direct move would be through the Barrier which is not allowed. A move
the Barrier, however, is possible which is, therefore allowed. Use the Man
there is a move route possible around the edge of the Barrier.

Manhattan distance is a heuristic function for calculating distance between
a grid. In the case of Dastan it is calculated by counting the sum of the nu
and then vertically (or vice versa) between a player starting location and th
in **Fig 2** below.

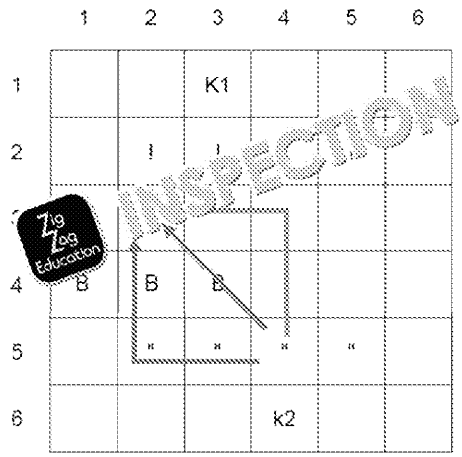
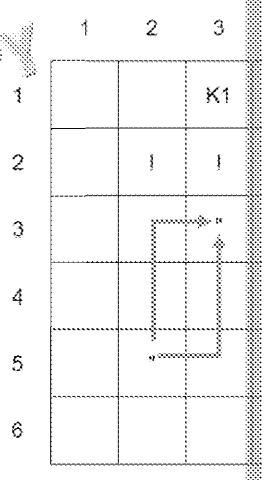


Fig 1



Fig

What you need to do

Task 1

i) Create a new class Barrier which sh ld inherit from the Square c
assigned an owner and gi h ymbol of a capital 'B' if it belong
lowercase 'b' if it b to player two.

ii) Cr ne public method ContainsBarrier in the Square class
has placed in that square.

(TASK CONTINUES ON THE NEXT PAGE)

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 2

i) Modify the CheckSquareIsValid method to check if the square bei
that a piece cannot occupy it or attempt to move it.

ii) Create a new method CheckBarrierIsValid in the Dastan class wh
a Barrier being placed by a player fits within the bounds of the boar
squares.

iii) Create a new method calle[illegible] Barrier in the Dastan class whic
Barrier onto the b[illegible] The Barrier will always be horizontal and the
centr[illegible] being asked where to place the Barrier.

Task 3

i) Create a new method called CheckManhattanDistance in the Das
paths from a starting square reference to a finishing square referen
starting row then down the finishing column and also down the star
finishing row. This is used to check if a selected move can traverse
over the top of it.

ii) Modify PlayGame to call CheckManhattanDistance which should
CheckPlayerMove used to set the value of the variable MoveLega

Note: This should be used for all moves even if they are too short to poten
may be able to go round. For a single or double move either horizontally o
should be considered; only for diagonal moves should you consider horizo
vertical and then horizontal.

Task 4

Test that t[illegible]nges you have made work:

- run the skeleton program.
- enter a position of 34 for the player one Barrier.
- enter a position of 42 for the player two Barrier.
- for player one: choose 9, then 1, then 1, then 24, then 46.
- for player two: choose 3, then 53, then 31.
- for player one: choose 2, then 25, then 45.
- for player two: choose 1, then 52, then 42, then 51.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing change[illegible]de to the PlayGa
- PROGRAM SOURCE CODE showing changes made to the Checks
CreatePieces methods in the Dastan class
- PROGRAM SOURCE CODE for the new private CheckBarrierIsVal
Che[illegible]hattanDistance methods in the Dastan class
- PROGRAM SOURCE CODE showing changes made to the Square
- PROGRAM SOURCE CODE showing the new Barrier class
- SCREEN CAPTURE(S) showing the required test

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Dastan

Programming Tasks (Extensi

Extension 1

Introduce a health scoring system for pieces. Each piece (except the Kotla) health points. Each time a piece is landed on, it incurs damage, reducing it each time a piece's health is reduced. When a piece reaches 0 health poin board. Only one piece can attack another at one time. When a piece is be player symbol should be shown on the left of the piece and the target piec the right of the square.

Damage is determined using this formula:

> *Position of move choice in the queue + Manhattan distance from the piece (number of rows different + number of columns different).*

Manhattan distance is a heuristic function for calculating distance between two locations, for example in a grid. In the case of Dastan it is calculated by counting the sum of the number of squares horizontally and then vertically (or vice versa) between a player starting location and the destination location as shown in **Fig 1**.

An attack using a move from position 1 in the move queue reduces health by 1 point. An attack from position 3 in the move queue reduces health by 3 points. This is how far away the opponent is from the attacker. This is the sum of the rows and columns. An attack from further away, therefore, incurs a greater level of damage.

Extension 2

Create a new game square called 'Qunbila Ghayr Muwajaha' (Unguided B has a 33% chance of appearing in any turn and is given to the current play a 10% chance of detonating. The player can either move away from the bo board. When the bomb is thrown the player can choose any board location location or a Kotla.

The 'Throw bomb' option should be available through the MoveOffer menu

If the bomb is thrown to an opponent's square, the opponent takes owners back or moves away from it. Each turn carries a 10% chance of the bomb d thrown to a Kotla square, the bomb loses ownership from either player an The bomb remains at this location until a player moves to the square conta up again and be able to throw it. Each turn carries a 10% chance of the bo

If a player piece is captured while holding the bomb, the ownership of the of the square.

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Extension 3

Introduce the concept of a 'Makinat Taftish' (Inspection Machine). This is a computer-controlled piece which does not belong to either player. After each player turn, the Inspection Machine should measure the distance from itself to all the other pieces on the board using Manhattan distance. The machine should then move itself towards the closest piece on the board, regardless of whether two pieces are the same distance away, the machine should select one at random. The machine can move in any direction, but only one square at a time.

The machine should repeat this behaviour once a turn until it reaches a player piece and captures it. Neither player gains any points for a piece being captured.

Manhattan distance is a heuristic function for calculating distance between two locations, for example in a grid. In the case of Dastan it is calculated by counting the sum of the number of squares horizontally and then vertically (or vice versa) between a player starting location and the finishing location as shown in **Fig 1**.

The machine does not place any weighting on a 'target' to move towards and can capture a player piece or a Kotla.

A player loses the game if their Kotla is captured by the Inspection Machine.

Extension 4

Introduce the concept of a 'Multi-Move'. This allows a player to combine two moves at a significant points cost.

Introduce a new option 9 to the main game playing menu called 'Multi-Move'. The player can select two move options to execute sequentially. The player will then move option 2, choosing a 'move to…' square reference for each option. The reference for move option 2 must be a legal move based on the 'move to…' of option 1. Both moves must be legal. The program should use error handling when entering illegal references and allow them to re-enter.

Selected moves in a multi-move can be from any position in `MoveOptionQueue` from the position of move in `MoveOptionQueue`.

On entering a legal multi-move, the game should move the selected player. The move should cost the player 2 points.

If the player lands on an opponent piece through either move 1 or move 2 the opponent piece should be captured as normal.

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Extension 5

Introduce the concept of a 'Khalad' – a mole. Introduce a new submenu op
move option from the main game playing menu. The submenu should offe
activate a 'mole' mode for the selected move option.

On selecting 'mole' mode, the move operates as norm[illegible] however, the pla
the board. A piece which is operating in 'mole' mode should be shown as a
for player 2, which is displayed on the left-hand side of each square instea
on the right-hand side of a square. This means that two pieces can occupy
'mole' mode and [illegible] the board 'surface'.

A piece in [illegible] mode can move around underneath the board using norm
be captured by an opponent piece on the surface of the board. Once the p
be in 'mole' mode, the submenu should change to now give the player the
piece after moving it. If a piece in 'mole' mode resurfaces in a square refer
piece, the current player captures that opponent piece. Once a player resu
piece, the 'mole' mode submenu should no longer be offered to the player.

A mole cannot move onto the Kotla square as the foundations are too dee

If an opponent also has a piece operating in 'mole' mode, one mole can ca
pieces on the board surface.

Extension 6

Introduce an option to 'preview a move' [illegible] to making it. Add a new optio
menu. On selecting this option, a player can select any move from position
valid player piece [illegible]. The player is then shown a 'preview' copy of
shows an [illegible] the squares which the selected move option can move t
current player piece.

The player should then be given the option to enter in a valid 'move to…' s
selected move option or go back to the main menu to choose a different m
valid 'move to…' square reference is selected, the game should make the

The player can 'preview a move' as many times as they like during the gam

The 'preview a move' option should not attempt to show the player 'move t
outside the bounds of the board.

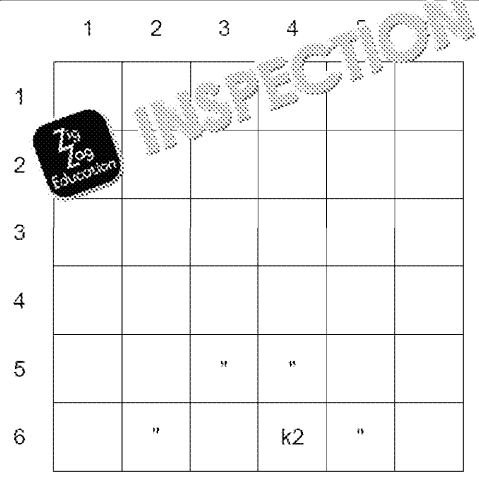
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Extension 7

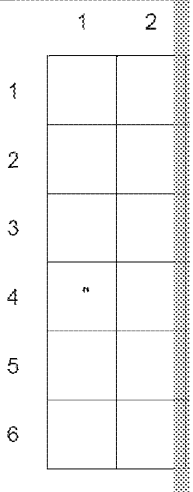
Introduce a new option at the start of the game to allow the players to plac
different formations prior to the game starting. Players can choose from an

All the positions are shown from the perspective of player 2.

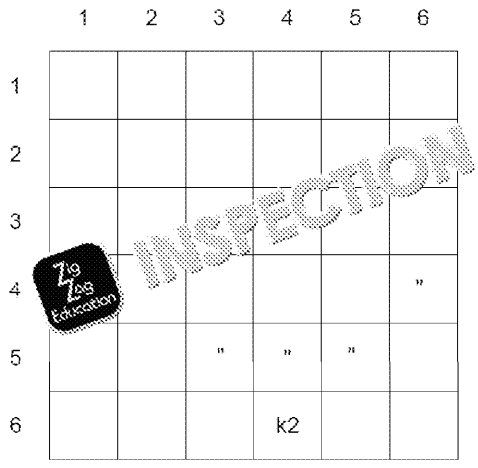
Once the players have selected their chosen starting positions, the game s



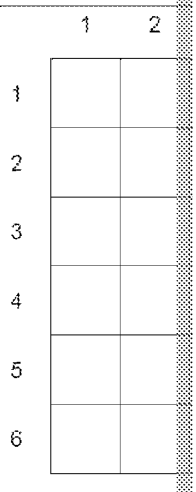
'Khandaq' Option (Trench)



'Iltifa



'Darba Rukniya' Option (Corner Kick)



'Khanj

Extension 8

Introduce the concept of an 'Al Amlaq' (Giant), wh rmed when a pla
of their **own** player pieces. A Giant is sh for player 1 and 'g' fo

Once a Giant has been cre by combining a player Mirza with a norma
and remains as a Gi the rest of the game.

A Giant ca e around the board using the same move options as a no
needs to land to within one square (in any direction) of any opposition piec
opponent Kotla and Mirza.

A Giant can be captured by any opponent piece as normal and is worth 20

INSPECTION COPY
COPYRIGHT PROTECTED

Extension 9

Introduce the concept of an 'Adra' (Chainmail). Add a new option 'C' to the
selecting this option, the player should be asked which piece they would li
player can only add chainmail to two pieces during the game. The chainm
clothes and therefore a piece's symbol doesn't change when it has the cha
forward-facing barrier which means that the piece cannot be attacked from
opponent piece can be one square in front of current player piece, it can
chainmail – it must approach the chainmail piece from either side or behind

A player can add chainmail to any two pieces in the game including the

Extension 10

Introduce the load and save features to the game. Add new options 'L' and
the player options to load a previously saved game or save the current gam

The load and save submenus should give the user the opportunity to enter
program should have appropriate error handling to prevent it from attempti
data or saving to an invalid location. The program should store appropriate
separated values to store all of the program data required to rebuild a gam
handling should be included when a game is being rebuilt to ensure that th
all valid within the bounds of the board.

Extension 11

Introduce a new feature that allows the size of the playing board and pieces
game, giving players the option to choose the size of the board. The dime
even; however, error handling should be included to prevent the board fro
larger than 10 × 10. (Appropriate formatting needs introducing on a 10 × 1
line up correctly.)

For boards of 6 to 8 columns wide, ensure that both player Kotlas are plac
and bottom rows of the board. A 7 column wide board should have 5 piece
wide board should have 6 pieces per player.

For boards 9 and 10 columns wide, introduce a second Kotla for each player on the appropriate top and bottom rows of the board. The Kotlas should be evenly distributed across the board. The player should still only have 1 Mirza, which should be placed in either of the Kotlas at random. A 9 or 10 column wide board should have 6 pieces per player, 3 of which should be in front of one Kotla and 3 in front of the other, as per the example shown.

	1	2	3
1			K
2		!	!
3			
4			
5			
6			
7			
8		"	"
9			K2

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Extension 12

Adjust the playing board to allow the sides to wrap around. On making a move, a player can move off the left- or right-hand side of the board and land on the correctly associated square on the opposite side of the board as if the board was wrapped around.

For example, a player can select a Cuirassier move for the piece in square 25 and move to square 31 which is one square forward followed by two squares to the left looking at the board from the point of view of the

Extension 13

Introduce the concept of an 'Muraqib' (Meerkat Lookout piece). At the start the opportunity to place their Muraqib on any empty square on the board. T represented by an 'M' symbol and player 2 Muraqib is represented by an 'm

The Muraqib is on constant lookout for the player it belongs to. For exampl legal move and the board and player 1 queue are updated, the player 2 Mu player 1 piece left on the board and test each of moves 1, 2, 3 from the pla see if it could threaten to capture any player 2 piece. If such a threat is pos player 2 in case they have missed that possible threat.

A Muraqib cannot be captured. If either player lands on the square contain disappears down into its burrow underneath the board. While it is in its bur it belongs to of any threatening moves from the player piece occupying t away from that square, the Muraqib should return to its lookout duties.

Extension 14

Introduce a new 'Aqrab' (Scorpion) option which can be added to any playe Aqrab can only applied to one piece per player. Once applied, the piece sy for a player 1 piece or '£' for a player 2 piece. A piece chosen to be a Aqra the board; however, when it is one square away from an opponent piece (i piece becomes paralysed and cannot move. This makes it vulnerable to b the Aqrab itself.

The Aqrab, however, can still be captured by any piece which can move fr squares away (in any direction). If the Aqrab moves away from a piece wit no longer paralysed and can move away as normal.

Extension 15

Introduce the concept of 'Quantum Mirza'. Add a new option 'Q' to the m options to allow the player to secretly swap their Mirza location with any of their other player p selecting the Quantum menu, the player should be asked for the board loc swap. The piece must belong to the current player. The program should us to ensure that a valid piece is selected. The program should then swap the Mirza locations. The player turn should then continue as normal.

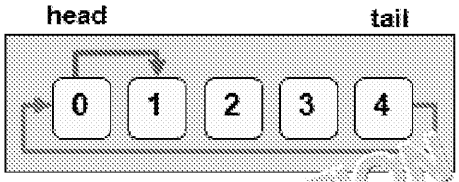
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Dastan

Exam-style Questions (Mark Sc

Q		Suggested Solution	Marks
1	(a)	$O(n^2)$	1 mark
	(b)	1 m[illegible]or [illegible]point • [illegible]atively efficient for smaller input sizes • … however, as the input size grows, the completion time increases • The rate of change is constantly changing using a quadratic function… • … which means that it does not scale up well	3 marks
2	(a)	1 mark for each point • You may mistype/misspell one of them • … which could mean that the code develops a logic error	2 marks
	(b)	1 mark for each point • One possible solution would be to define them as constants (1 mark) • … which would mean that you would get an error with an undefined identifier before running the program	2 marks
3	(a)	Every square in the board is treated as a square [1 mark] but some of them may be Kotlas (which inherit from Square) [1 mark] so will execute the overridden method on the Kotla due to polymorphism because although treated as a Square will behave as a Kotla [1 mark]	3 marks
4	(a)	Because the position of player one's Kotla is determined by the number of columns DIV 2 which gives 3 [1 mark] and the position of player two's Kotla is determined by the number of columns DIV 2 and then add 1 which gives 4 [1 mark]	2 marks
	(b)	Change the calculation for player one [1 mark] to (NoOfColumns+1) DIV 2 [1 mark] which will round up for odd numbers [1 mark] but round down for even numbers [1 mark]	4 marks
5	(a)	As the Direction attribute is part of the Player class [1 mark] both of these methods could go modify the NewMoveOption when it is received in the AddToMoveOptionQueue and UpdateMoveOptionQueueWithOffer methods [1 mark] to modify each non-zero value for RowChange and ColumnChange by multiplying it by the Direction for the current player [1 mark]	3 marks
6	(a)	A queue is more appropriate because move options are added to the back of the queue but could not be added to the bottom of the stack as it is a LIFO structure [1 mark] and removed from the front of the queue because it is a FIFO structure [1 mark]	2 marks

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Q		Suggested Solution	Marks
6	(b)	A circular queue would need a head variable [1 mark] and a tail variable [1 mark]…  … so that when an item is added to the queue, the rear pointer could be incremented or wrapped back around to 0 if it was greater than 4 [1 mark] and when an item is removed from the queue the head pointer could be increased or wrapped back around to 0 if it was greater than 4 [1 mark]	4 marks
7	(a)	Each square is referred to by a two-digit number, the method extracts the first digit using MOD, subtracts one [1 mark] and then multiplies it by number of columns [1 mark], then extracts the second digit of the square reference using DIV, subtracts one and adds the two together. [1 mark]	3 marks
8	(a)	One dimension could be the row [1 mark] and the second dimension could be the column [1 mark]	2 marks
	(b)	An array is static so the amount of memory used will not change and the board size is fixed so this is appropriate	1 mark
9	(a)	Metadata describes the data in a file [1 mark] Possible examples (any sensible answer will do): • Board size (resolution) • Number of pieces for each player	2 marks
10	(a)	This is not polymorphism because each of the five methods creates a MoveOption object [1 mark] which is the same class but contains different data [1 mark]. In order to be polymorphism you need to have child classes being treated as their parent which is not the case here [1 mark].	2 marks
	(b)	This is polymorphism because each of the five different MoveOption methods (e.g. ChowkidarMoveOption) for each move inherits from MoveOption and so can be treated as a MoveOption [1 mark] but will actually behave as themselves [1 mark] meaning that you could still have a collection of MoveOptions, all of which would actually be children of MoveOption [1 mark]	2 marks
11	(a)	1 mark for each point • super() is used to refer to the base class object • And call a method within it	2 marks
	(b)	Overriding	1 mark
	(c)	1 mark for each point • polymorphism uses **multiple** implementations • of a method with the same name • by selecting which version to run based on the number and type of parameters passed • within the same class definition	3 marks

INSPECTION COPY
COPYRIGHT PROTECTED

Q		Suggested Solution	Marks
12	a)	1 mark for each point • A priority queue has different points at which items can join the queue according to priority • They join at the back of the section according to their priority, almost like sub-queues • If there are no items queued in the corr[illegible]ity section then they join the queue at the [illegible] of the next lower priority or at the ba[illegible] next higher • Items are still t[illegible] one at a time from the front of the entire q[illegible] join at the back of the appropriate sub-[illegible]section	4 marks
13	(a)	1 m[illegible] each point • The name describes the purpose of the variable • which makes the code easier to read/understand/follow	2 marks
14	(a)	1 mark for each point • It can be accessed by children/subclasses • and from within the class itself	2 marks
	(b)	It can be accessed from anywhere	1 mark
	(c)	It can only be accessed from within the class	1 mark
	(d)	They allow correct encapsulation [1 mark] of the class… which means that you can only interact with the class through the intended interface [1 mark]… but it still allows for d[illegible]ct access within the class where required [1 mark]. Also avoids exposing attributes and methods that are either dangerous to expose or unnecessary outside the class [1 mark].	3 marks
15	(a)	1 mark for each [illegible] • [illegible]ion returns a whole number (and a remainder) • floating point division returns a decimal value with a decimal point	2 marks
	(b)	It has two values, true or false	1 mark

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Dastan

Programming Tasks (Mark Sch

Task 1

Coding:

- Create a new method C[illegible]reateCustomPlayers which allows the user to enter in [illegible] continue u[illegible]e [illegible]re different. [1 mark]

Example Solu[illegible]

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 differ
        self._Players.appe[illegible]("Player Two", -1))
    #END ADDITION
```

Testing:

Display an appropriate error message if the user enters in two matching names. Co[illegible] 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: Vict

   1  2  3  4  5  6
 --------------------
1|  |  |K1|  |  |  |
2|  | || || || ||  |
3|  |  |  |  |  |  |
4|  |  |  |  |  |  |
5|  | "| "| "| "|  |
6|  |  |  |k2|  |  |
 --------------------

Move option offer: jazair

Tom
Score: 100
Move option[illegible]: 1. ryott   2. chowkidar   3. cuirassier   4. fau

Turn: Tom

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

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 2

Coding:

- Adding a new MoveOptionOffer to the CreateMoveOptionOffer method, a
  CreateMoveOption method, and adding the move option to both players in the
  parameter set correctly. [1 mark]
- Adding a 'faris' to the CreateMoveOption method
  method. [1 mark]
- Create a new method CreateFari
  the possible positions for the F [1 mark]

Example So

Changes to C MoveOptionOffer:

```
    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(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
```

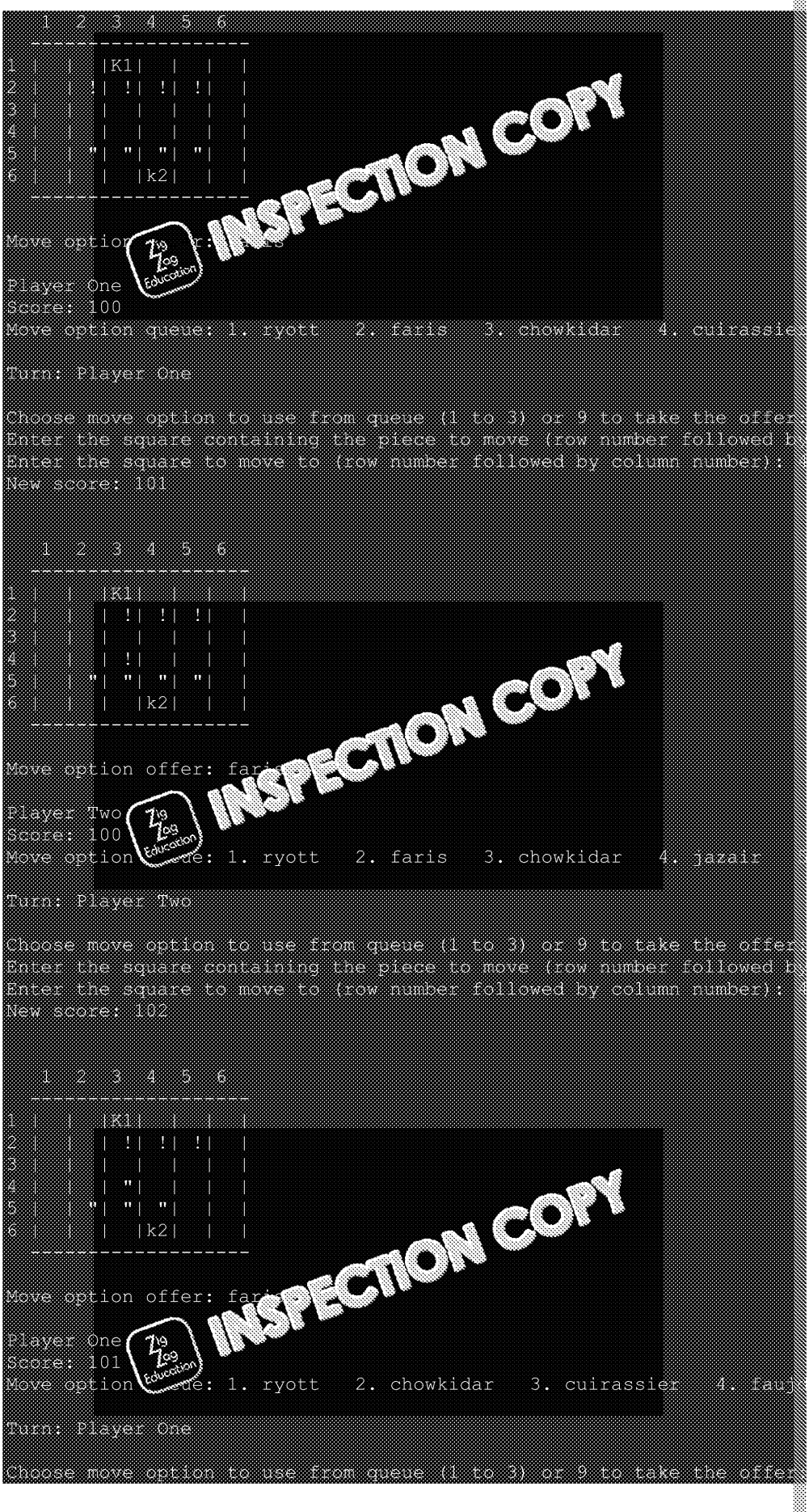
Changes to CreateMoveOptions:

```
    def __CreateMove(self):
        self.[illegible].AddToMoveOptionQueue(self.__CreateMove
        se Players[0].AddToMoveOptionQueue(self.__CreateMove
        sel Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[1].AddToMoveOptionQueue(self.__CreateMove
        self._Players[1].AddToMoveOptionQueue(self.__CreateMove
```

INSPECTION COPY
COPYRIGHT PROTECTED

Testing:

Displaying the Faris move option correctly in the player one queue and moving a pl
legal Faris move. [1 mark]



INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Task 3

Coding:

- Adding a new `MoveOptionOffer` to the `CreateMoveOptionOffer` method, a
  `CreateMoveOption` method, and adding the move option to both players in the
  parameter set correctly. [1 mark]
- Adding a Sarukh to the `CreateMoveOption` meth a w option which calls t
  method. [1 mark]
- Create a new method `CreateSa t v Option` which correctly uses the Dir
  the possible positions for h move. [1 mark]

Example So

Changes to C eMoveOptionOffer:

```
    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._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[0].AddToMoveOptionQueue(self.__CreateMove
        self._Players[1].AddToMoveOptionQueue(self.__CreateMove
        self._Players[1].AddToMoveOptionQueue(self.__CreateMove
```

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(1 * Direction, 1)
        NewMoveOption.AddToPossibleMoves(NewMove)
        NewMoveOption.AddToPossibleMoves(NewMove)
        return NewMoveOption
    #END ADDITION
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Testing:

Displaying the Sarukh move option correctly in the player one queue and moving a
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. cuirassi

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
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

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer
Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
New score: 101


   1  2  3  4  5  6
  ------------------
1 |  |  |K1|  |  |  |
2 |  |  | !| !| !|  |
3 |  |  |  |  |  |  |
4 |  | !|  |  |  |  |
5 | "|  | "| "| "|  |
6 |  |  |  |k2|  |  |
  ------------------

Move option offer:

Player One
Score: 101
Move option queue: 1. ryott   2. chowkidar   3. cuirassier   4. fau

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offe
```

INSPECTION COPY

COPYRIGHT PROTECTED

Task 4

Coding:

- Change PlayGame to randomly award a Wafr to the current player and if one has 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 points cost to the player. [1 mark]
- Create a new method AwardWafr in the Dastardly class which has a 25% chance
- Adding the WafrAwarded attribute to Player with get/set methods for WafrAwa

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)
    #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.G
                print("You have been awarded a Wafr, you can se
                queue for free this turn.")
                self._CurrentPlayer.SetWafrAwarded()
                Wafr = True
            if Wafr:
                while Choice < 1 or Choice
                    Choice = int(input("Choose move option to us
            else:
                while Choice < 1 or Choice > 3:
                    Choice = int(input("Choose move option to u
                    to take the offer: "))
                    if Choice == 9:
                        self.__UseMoveOptionOffer()
                        self.__DisplayState()
            #END CHANGE
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

```
            if MoveLegal:
                PointsForPieceCapture = self.__CalculatePieceCapture
                #CODE CHANGE
                if not Wafr:
                    self._CurrentPlayer.ChangeScore(-(Choice + (
                #END CHANGE
                self._CurrentPlayer.UpdateQueueAfterMove(Choice)
```

Testing:

Show a player being awarded a wafr and selecting a move from position 4 or 5 in th incurring a cost. [1 mark]

```
Move option queue: jazair

Player Two
Score: 104
Move option queue: 1. chowkidar   2. jazair   3. faujdar   4. cuiras

Turn: Player Two

You have been awarded a Wafr, you can select any move from your queu
Choose move option to use from queue (1 to 5): 5
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
New score: 109


   1  2  3  4  5  6
  -------------------
1 |  |  |K1|  |  |  |
2 |  | !| !| !| !|  |
3 |  | "|  |  |  |  |
4 |  |  |  |  |  |  |
5 |  |  | "| "| "|  |
6 |  |  |  |k2|  |  |
  -------------------
```

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 5

Coding:

- Change `PlayGame` to give new menu option 8 and reduction of player score by 5
- Adding the `Opponent` variable (or similar) to `PlayGame` and correctly assigning [1 mark]
- Correctly printing out the opponent's queue. [1 mark]
- Creation of `GetJustQueue` which calls the `GetQueueAsString` method for the `Player`. [1 mark]

Example Solution

Changes to `PlayGame`:

```
            while Choice < 1 or Choice > 3:
                Choice = int(input("Choose move option to use f
                see your opponent's queue or 9 to take the offe
                if Choice == 9:
                    self.__UseMoveOptionOffer()
                    self.__DisplayState()
                #CODE ADDED
                elif Choice == 8:
                   Opponent = self._Players[1] if self._CurrentPla
                   else self._Players[0]
                    print(Opponent.GetJustQueue())
                    self._CurrentPlayer.ChangeScore(-5)
                    print("New score: " + str(self._CurrentPlaye
                #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

```
Player One
Score: 100
Move option queue: 1. ryott   2. chowkidar   3. cuirassier   4. fau

Turn: Player One

Choose move option to use from queue (1 to 3), 8 to see your opponen
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 opponen
offer:
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 6

Coding:

- Create a new method `GetValidInt` which returns true if the user enters in a va
suitable message and force the user to retry until they have entered a valid integ
- Change `PlayGame` to use the `GetValidInt` method on the main game menu to
the move queue choice. [1 mark]
- Change `GetSquareReference` to use the `GetValidInt` method for choosing a
suitable error message only allow valid numeric input. [1 mark]
- Change `UseMoveOptionOffer` to use the `GetValidInt` method for choosing w
allow valid integer input to include a range of 1 to 5 to prevent an invalid queue
should include a valid error message and a prompt to re-enter until it is valid. [1

A: passing of prompt to `GetValidInt()` instead, but do not award if the line sp

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,
        return number
    #END ADDITION
```

Changes to `PlayGame`:

```
            while Choice < 1 or Choice > 3:
                #CODE CHANGE
                print("Choose move option to use from queue (1
                offer: ",end="")
                Choice = self.__GetValidInt()
                #END CHANGE
                if Choice == 9:
                    self.__UseMoveOptionOffer()
                    self.__DisplayState()
            while not SquareIsValid:
                StartSquareReference = self.__GetSquareReferenc
                move")
                SquareIsValid = self.__CheckSquareIsValid(Start
                #CODE ADDED
                if not SquareIsValid:
                    print("You must ent[illegible]id square.")
                #END ADDITION
            SquareIsValid = Fal
            while not SquareIsValid:
                FinishSquareReference = self.__GetSquareReferen
                SquareIsValid = self.__CheckSquareIsValid(Finis
                #CODE ADDED
                if not SquareIsValid:
                    print("You must enter a valid square.")
                #END ADDITION
            MoveLegal = self._CurrentPlayer.CheckPlayerMove(Cho
            FinishSquareReference)
```

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Changes to GetSquareReference:

```
    def __GetSquareReference(self, Description):
        #CODE CHANGE
        print("Enter the square " + Description + " (row number fo
        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 replac
        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(Repl
        self.__CreateMoveOption(self._MoveOptionOffer[self._Move
        self._CurrentPlayer.GetDirection()))
        self._CurrentPlayer.ChangeScore(-(10 - (ReplaceChoice *
        self._MoveOptionOfferPosition = random.randint(0, 4)
```

Testing:

Display an appropriate error message if the user enters a non-valid inputs for the n
to place MoveOptionOffer in the queue.

```
   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. fauj

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Invalid input, you must enter an integer, please try again: 1
Enter the square containing the piece to move (row number followed b
You must enter a valid square.
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
New score: 104

   1  2  3  4  5  6
 ---------------------
1|  |  |K1|  |  |  |
2|  |  | !| !| !|  |
3|  | !|  |  |  |  |
4|  |  |  |  |  |  |
5|  | "| "| "| "|  |
6|  |  |  |k2|  |  |
 ---------------------
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

```
Move option offer: jazair

Player Two
Score: 100
Move option queue: 1. ryott   2. chowkidar   3. jazair   4. faujdar

Turn: Player Two

Choose move option to use from queue (1 to 4) or 5 to take the offer
Choose the move option from your queue to exchange (1 to 5): 8
You must enter a number from 1-5
Please re-enter your selection:
```

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Task 7

Coding:

- Adding the ChoiceOptionsLeft attribute to Player with getter method. Initialis
- Create a new method DecreaseChoiceOptionsLeft in Player which decrem
  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,
- Change UserMoveOptionOffer to call DecreaseChoiceOptionsLeft for the
  a move option from the menu. [1 mark]

Example So

Changes to P

```
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() >
                    Choice = int(input("Choose move option to us
                    to take the offer: "))
                else:
                #END ADDITION
                    Choice = int(input("Choose move option to u
                if self._CurrentPlayer.GetChoiceOptionsLeft() >
                    self.__UseMoveOptionOffer()
                    self.__DisplayState()
```

Changes to UseMoveOptionOffer:

```
    def __UseMoveOptionOffer(self):
        ReplaceChoice = int(input("Choose which move option from
        (1 to 5): "))
        self._CurrentPlayer.UseMoveOptionQueueWithOffer(Repl
        self.__CreateMoveOption(self._MoveOptionOffer[self._Move
        self._CurrentPlayer.GetDirection()))
        self._CurrentPlayer.ChangeScore(-(10 - (ReplaceChoice *
        self._MoveOptionOfferPosition = random.randint(0, 4)
        #CODE ADDED
        self._CurrentPlayer.DecreaseChoiceOptionsLeft()
        print(f"You have {self._CurrentPlayer.GetChoiceOptionsL
        left.")
        #END ADDITION
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Testing:

Show player one selecting a move from the move option offer menu and decreasing

INSPECTION COPY

COPYRIGHT PROTECTED

```
Turn: Player One

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

Player One
Score: 82
Move option queue: 1. jazair   2. ryott   3. faujdar   4. faujdar

Turn: Player One

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

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Task 8

Coding:

- Writing the ResetQueueBackAfterUndo method which calls the ResetQueueB
  (the same one it was passed - Position – but adjusted by -1 to make it an inde
  pops the item from the end of the queue and returns it to its original position. [1
- Asking the player if they would like to undo after the m[illegible] played their move an
- Correctly handling the undo to deduct 5 [illegible] and reset the board and queue.

Example Solution

Changes to [illegible]:

```
    #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.
    #END ADDITION
```

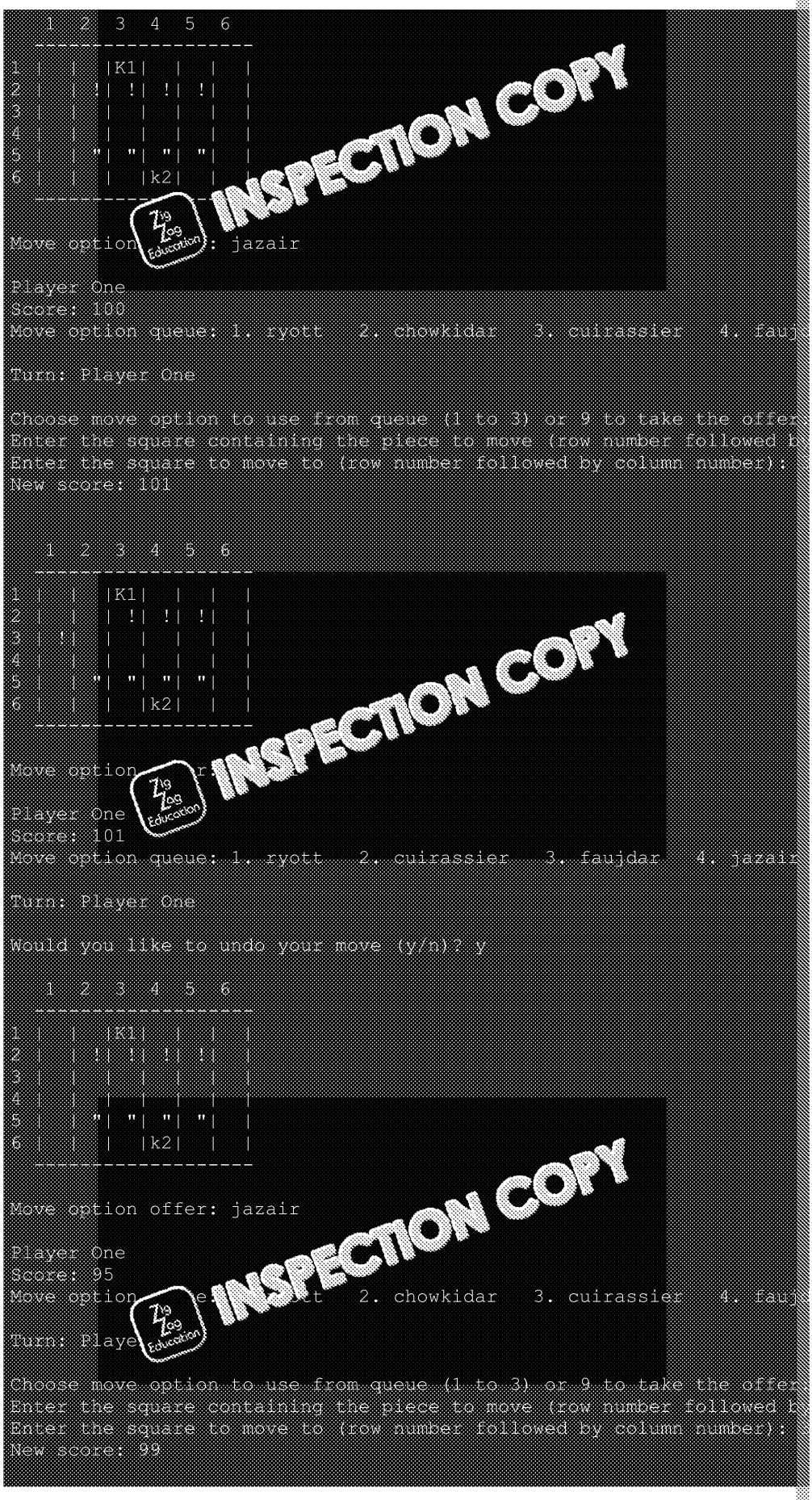
Changes to PlayGame:

```
            MoveLegal = self._CurrentPlayer.[illegible]PlayerMove(Cho
            FinishSquareReference)
            #CODE CHANGE
            undo = "n"
            if Move[illegible]
                [illegible]sScore = self._CurrentPlayer.GetScore()
                PointsForPieceCapture = self.__CalculatePieceCaptur
                self._CurrentPlayer.ChangeScore(-(Choice + (2 *
                self._CurrentPlayer.UpdateQueueAfterMove(Choice
                self.__UpdateBoard(StartSquareReference, Finish
                self.__UpdatePlayerScore(PointsForPieceCapture)
                print("New score: " + str(self._CurrentPlayer.G
                self.__DisplayState()
                undo = input("Would you like to undo your move
                if undo.strip().lower() == "y":
                    self.__UpdateBoard(FinishSquareReference,St
                    self._CurrentPlayer.ResetQueueBackAfterUndo
                    self._CurrentPlayer.ChangeScore(PreviousScore-se
                else:
                    undo = "n"
            if undo == "n":
                if self._CurrentPlayer.[illegible](self._Players[0])
                    self._Curre[illegible] = self._Players[1]
                else:
                    [illegible]CurrentPlayer = self._Players[0]
            [illegible]
            [illegible]Over = self.__CheckIfGameOver()
```

INSPECTION COPY
COPYRIGHT PROTECTED

Testing:

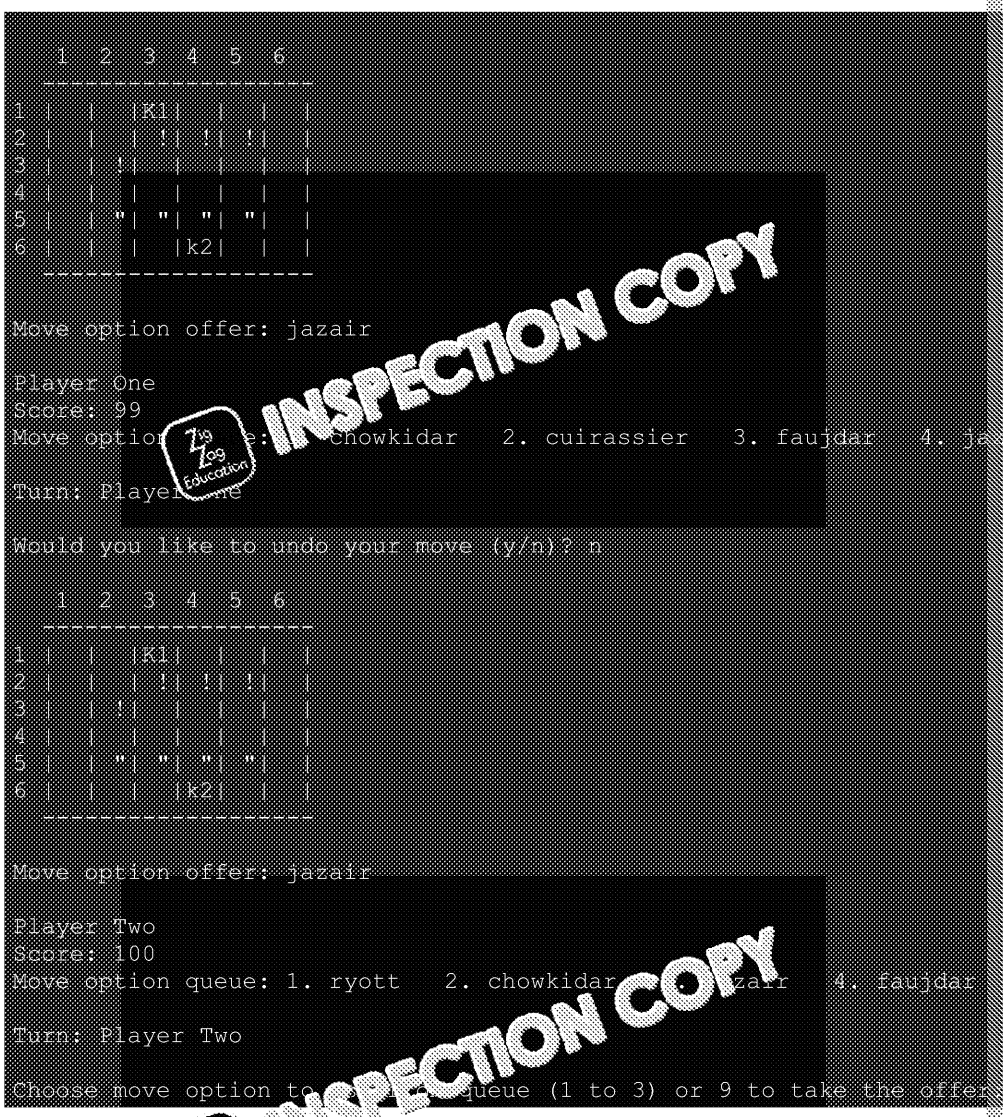
- Showing that a move can be undone and that 5 points are deducted. [1 mark]
- Showing that the same player can still play their turn and that the game can con
  [1 mark]



```
   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. fauj

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
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

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. fauj

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
New score: 99
```

INSPECTION COPY

COPYRIGHT PROTECTED



INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Task 9

Coding:

- `CreateMoveOptionOffer` method has been modified to the append "sahm" as dealing with the `Name` parameter of "sahm" in the `CreateMoveOption` method.
- 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 `CalculateSahmMove`). [1 mark]
- Correctly awarding points for all removed/destroyed pieces (even if a piece was `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.AddToPossibleMove(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).G
    #END ADDITION
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Changes to PlayGame:

```
            SquareIsValid = self.__CheckSquareIsValid(Start
        #CODE CHANGE
        if self._CurrentPlayer.ChoiceIsSahm(Choice):
            if self._CurrentPlayer.GetSahmUsed():
                print("You have already used your Sahm!")
            else:
                self._CurrentPlayer.SetSahmUsed()
                PointsForPieceCapture = self.__CalculateSahm
                self._CurrentPlayer.ChangeScore(-(Choice +
                self._CurrentPlayer.UpdateQueueAfterMove(Ch
                self.__UpdatePlayerScore(PointsForPieceCapt
                print("New score: " + str(self._CurrentPlay
        ;
            SquareIsValid = False
            while not SquareIsValid:
                FinishSquareReference = self.__GetSquareRef
                SquareIsValid = self.__CheckSquareIsValid(Fi
            MoveLegal = self._CurrentPlayer.CheckPlayerMove(
            StartSquareReference, FinishSquareReference)
            if MoveLegal:
                PointsForPieceCapture =
                self.__CalculatePieceCapturePoints(FinishSq
                self._CurrentPlayer.ChangeScore(-(Choice +
                self._CurrentPlayer.UpdateQueueAfterMove(Ch
                self.__UpdateBoard(StartSquareReference, Fi
                self.__UpdatePlayerScore(PointsForPieceCapt
                print("New score: " + str(self._CurrentPlay
        #END CHANGE
        if self._CurrentPlayer.Same(self._Players[0]):
```

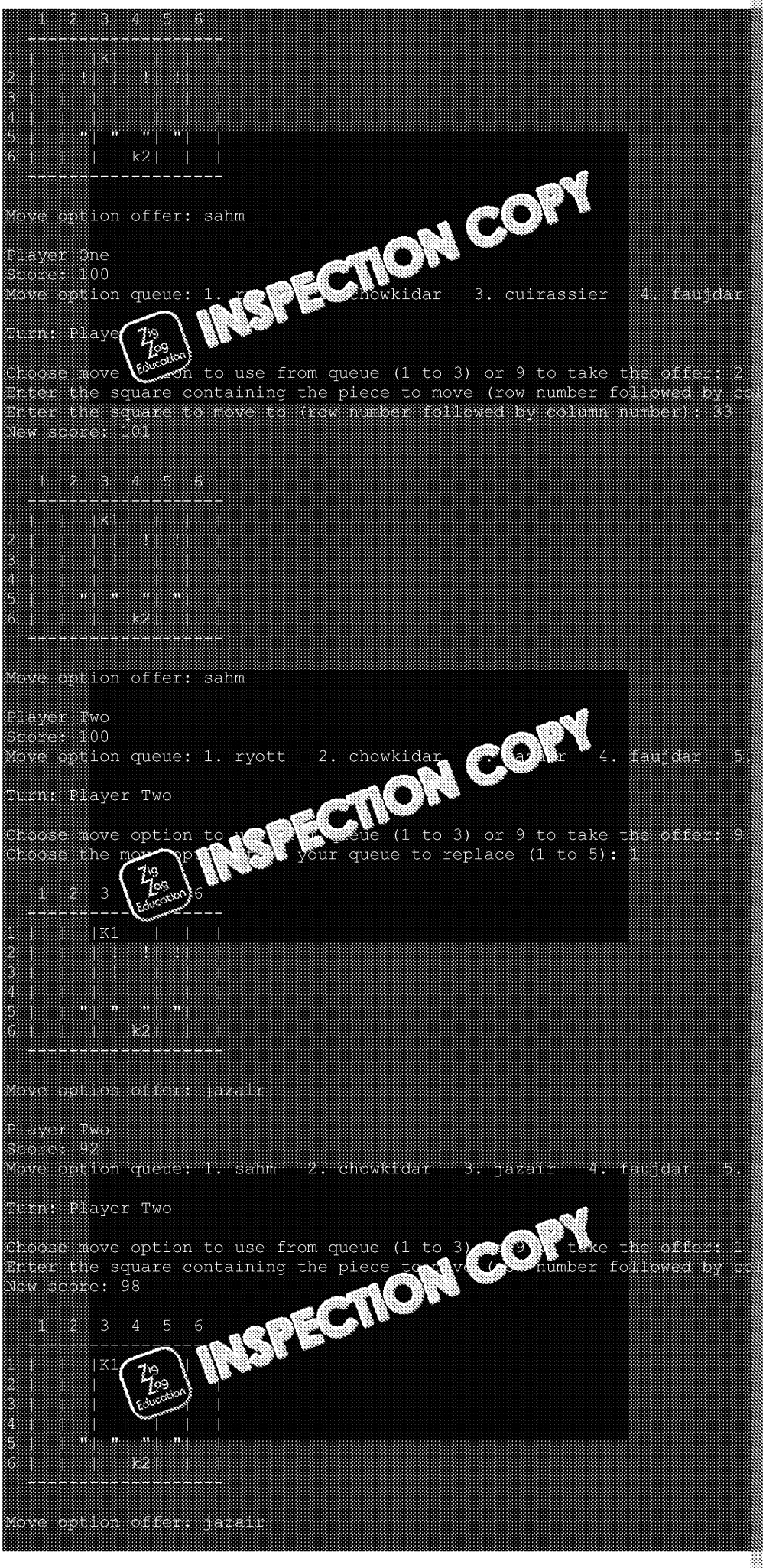
Code for CalculateSahmMove:

```
    #CODE
    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(1
            if BoardSquare.GetPieceInSquare() is not None and not
                Score += BoardSquare.GetPieceInSquare().GetPoint
                BoardSquare.RemovePiece()
        return Score
    #END ADDITION
```

Testing:

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

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education



INSPECTION COPY

COPYRIGHT PROTECTED

```
Player One
Score: 101
Move option queue: 1. ryott   2. cuirassier   3. faujdar   4. jazair   5.

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer:
  ---------------------
1 |   |   |K1|   |   |  |
2 |   | ! | !| !| !|  |
3 |   |   |  |  |  |  |
4 |   |   |  |  |  |  |
5 |   | " | "| "| "|  |
6 |   |   |  |k2|  |  |
  ---------------------

Move option offer: ryott

Player One
Score: 100
Move option queue: 1. ryott   2. chowkidar   3. cuirassier   4. faujdar

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 co
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. raaket   2. chowkidar   3. jazair   4. faujdar   5.

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.

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 co
New score: 9
```

INSPECTION COPY

COPYRIGHT PROTECTED

Task 10

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
- Creating a Kotla of the correct type in the square and re[illegible]g the piece. [1 ma

Example Solution

Changes to `PlayGame`:

```
            while Choice < 1 or (Choice > 3 and Choice != 7):
                Choice = int(input("Choose move option to use f
                create a Kotla or 9 to take the offer: "))
                if Choice == 9:
                    self.__UseMoveOptionOffer()
                    self.__DisplayState()
            if Choice==7:
                while not SquareIsValid:
                    StartSquareReference = self.__GetSquareRefe
                    to sacrifice for a new Kotla")
                    SquareIsValid = self.__CheckSquareIsValid(S
                if self._CurrentPlayer.SameAs(self._Players[0])
                    self._Board[self.__GetIndexOfSquare(StartSq
                    Kotla(self._CurrentPlayer, "K")
                else:
                    self._Board[self.__GetIndexOfSquare(StartSq
                    Kotla(self._CurrentPlayer, "k")
                self._CurrentPlayer
            else:
                while not SquareIsValid:
                    StartSquareReference = self.__GetSquareRefe
                    move")
                    SquareIsValid = self.__CheckSquareIsValid(S
                SquareIsValid = False
                while not SquareIsValid:
```

Testing:

Showing the creation of the new Kotla (even if the letter is wrong) and removal of th

```
   1  2  3  4  5  6
 --------------------
1|  |  |K1|  |  |  |
2|  | !| !| !| !|  |
3|  |  |  |  |  |  |
4|  |  |  |  |  |  |
5|  | "| "| "| "|  |
6|  |  |  |k2|  |  |
 --------------------

Move option offer: jazair

Player One
Score: 100
Move option [illegible]: 1. ryott   2. chowkidar   3. cuirassier   4. fau

Turn: Player One

Choose move option to use from queue (1 to 3), 7 to create a Kotla o
Enter the square containing the piece to sacrifice for a new Kotla (
number): 52
```

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Task 11

Coding:

- Adding option 6 to the menu which brings up a list of options to modify the queue
- Option a correctly reverses the player's queue in a method inside MoveOptionQ
- Option b correctly swaps queue with the opponent without breaking encapsulatio 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 option chosen to the front of the queue and shuffles up th
- Option e quits and does not cost any points but the other options all cost 3 points

Example Sol

Changes to PlayGame:

```
            while Choice < 1 or Choice > 3:
                Choice = int(input("Choose move option to use f
                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 qu
        print("a) Reverse your queue")
        print("b) Swap queues with opponent")
        print("c) Swap the first and last move options in your
        print("d) Move option of your choice to position 1 i
        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 po
            the front(2-5): "))
        self._CurrentPlayer.ChangeScore(-3)
        self._DisplayState()
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

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 P[illegible]:

```
    #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)
    #END ADDITION
```

Testing:

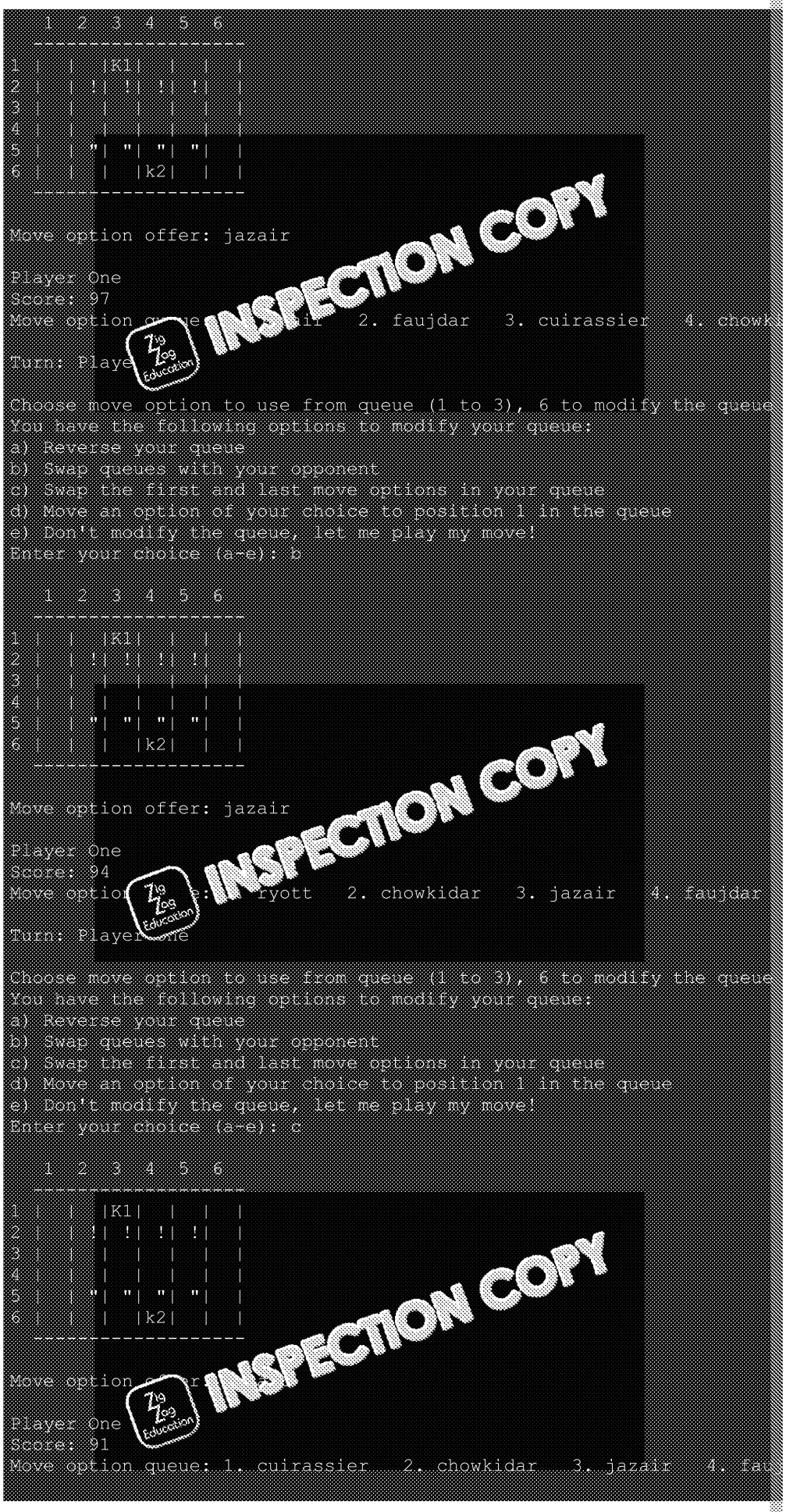
- Showing [illegible] of options a–d working. [1 mark]
- Showing t[illegible]aining three options working. [1 mark]
- Showing option e and the scoring working correctly. [1 mark]

```
   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. fauj

Turn: Player One

Choose move option [illegible] from queue (1 to 3), 6 to modify the queue
You have th[illegible] options to modify your queue:
a) Reverse [illegible]ueue
b) Swap que[illegible]ith 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
```

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education



INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

```
Turn: Player One

Choose move option to use from queue (1 to 3), 6 to modify the queue
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 (2-6): 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. ja

Turn: Player One

Choose move option to use from queue (1 to 3), 6 to modify the queue
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
```

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 12

Coding:

- Creating and storing the number of pieces correctly in the new protected attribut
- Adding a call to `CheckReincarnation` in the correct place. [1 mark]
- Creating `CountNormalPieces` to correctly return the numbe of pieces excludin [1 mark]
- Correctly detecting when a piece reaches the opposition's back row. [1 mark]
- Having a condition to only allow reincarnation if the player has fewer pieces than
- Correctly handling the reincarnation in the player's own back row and checking

Example Solu

Changes to D

```
    def __init__(self, R, C, NoOfPieces):
        self._NoOfPieces = NoOfPieces #LINE ADDED
```

Changes to PlayGame:

```
            if MoveLegal:
                self.__CheckReincarnation(FinishSquareReference
```

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._Cu
                PieceInSquare.GetPiece() == "piece":
                    Pieces
        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.__CountNormalPiec
                print("Congratulations, you have earned a reinc
                ReincarnationCol = int(input("Which column woul
                reincarnated on? "))
                while self._Board[self.__GetIndexOfSquare(10+Re
                .GetPieceInSquare() != None:
                    print("The square must be empty.")
                    ReincarnationCol = int(input("Which column
                    to be reincarnated on? "))
                self._Board[self.__GetIndexOfSquare(10+Reincarn
                ("piece", self._Players[0], 1, "'"))
        else:
            if Row == 1 and self.__CountNormalPieces() < self._N
                print("Congratulations, you have earned a reinc
                ReincarnationCol = int(input("Which column woul
                reincarnated on? "))
                while self._Board[self.__GetIndexOfSquare(6*10+
                .GetPieceInSquare() != None:
                    print("The square must be empty.")
                    ReincarnationCol = int(input("Which column
                    to be reincarnated on? "))
                self._Board[self.__GetIndexOfSquare(6*10+Reinca
                ("piece", self._Players[1], 1, '"'))
    #END ADDITION
```

INSPECTION COPY
COPYRIGHT PROTECTED

Testing:

Correctly showing the moves as requested in the tests, specifically including the ch
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 queue: 1. ryott   2. chowkidar   3. cuirassier   4. fauj

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
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 queue: 1. ryott   2. chowkidar   3. jazair   4. faujdar

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
New score: 104
```

```
   1  2  3  4  5  6
 ---------------------
1 | "|  |K1| !|  |  |
2 |  | !|  !|  |  |  |
3 |  |  |  |  |  |  |
4 |  |  |  |  |  |  |
5 |  | "|  "|  "|  |  |
6 | !|  |  |k2|  |  |
 ---------------------

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

Turn: Player One

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

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Task 13

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 ha
- Resetting the CampedTurns attribute if the square becomes empty or changes
- Allowing the player to make a move that costs zero p[illegible]w[illegible]en they have held
- Showing the correct A and a symbols when t[illegible]e Taz[illegible] occupied by overriding
- Correctly resetting the symbol for th[illegible]Taziz to 'x' when a player leaves by overri

Example Solution

Changes to (illegible) Board:

```
            #CODE ADDED

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

                S = Taziz()
            #END ADDITION
            else:
                S = Square()
```

Changes to PlayGame:

```
    def PlayGame(self):
        GameOver = False
        taziz = self._Board[self.__GetIndexOfSquare((self._NoOf
        (self._NoOfColumns // 2 + self._NoOfCo[illegible]s % 2))] #LINE A
        while not GameOver:
```

```
            if MoveIsLegal:
                PointsForPieceCapture = self.__CalculatePieceCaptur
                #START CHANGE
                self._CurrentPlayer.UpdateQueueAfterMove(Choice)
                self.__UpdateBoard(StartSquareReference, Finish
                taziz.CheckCamp()
                if not taziz.GetCampedTwoTurns():
                    self._CurrentPlayer.ChangeScore(-(Choice +
                #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._Piece[illegible] != None:
            if (se[illegible]ceInSquare.GetBelongsTo().GetDirection()
            S[illegible]ol == "A") or (self._PieceInSquare.GetBelong
            a[illegible]lf._Symbol == "a"):
                self._CampedTurns += 1

    def GetCampedTwoTurns(self):
        return self._CampedTurns >= 2
```

INSPECTION COPY
COPYRIGHT PROTECTED

```
    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]

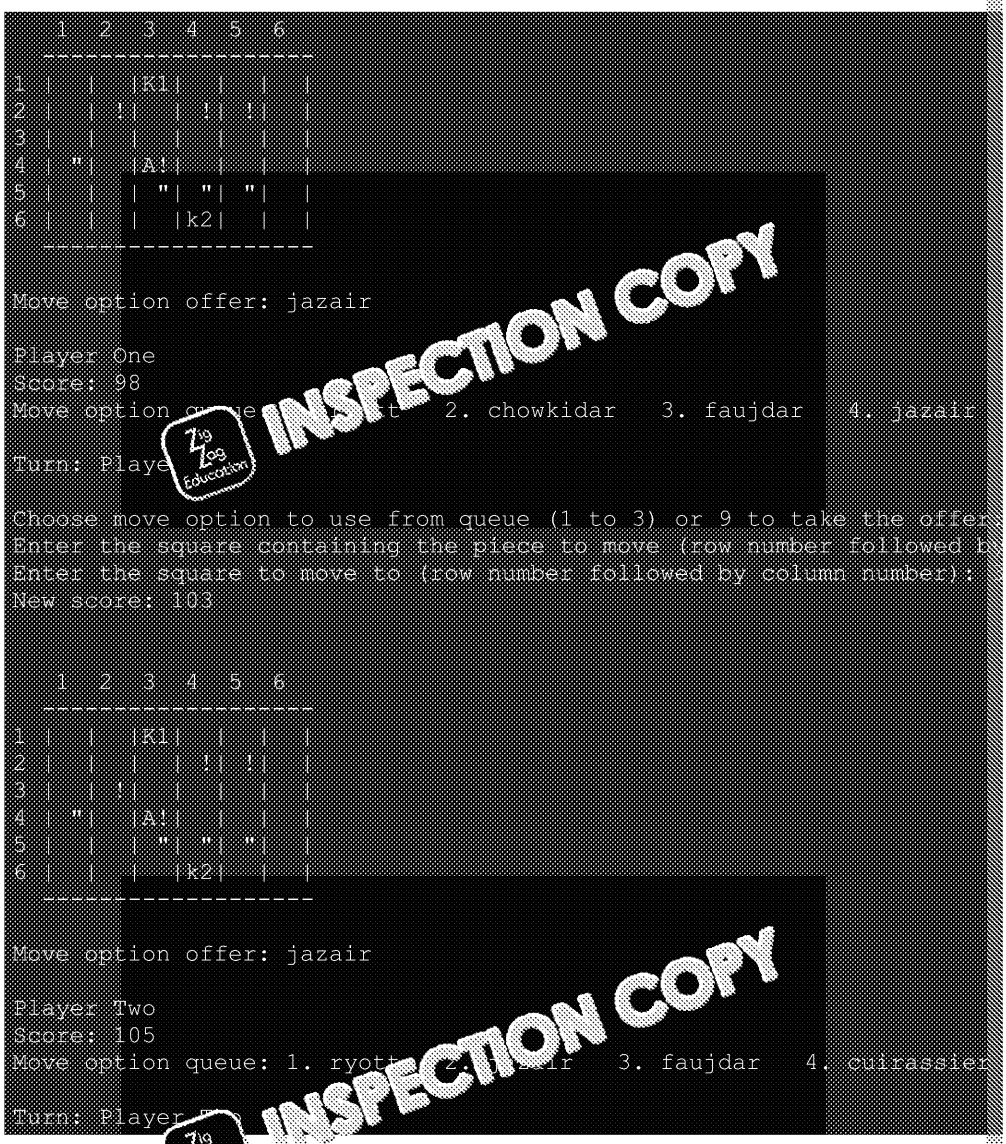
```
   1  2  3  4  5  6
  -------------------
1 |  |  |K1|  |  |  |
2 |  | !| !| !| !|  |
3 |  |  |  |  |  |  |
4 |  |  |x |  |  |  |
5 |  | "| "| "| "|  |
6 |  |  |  |k2|  |  |
  -------------------

Move option offer: jazair

Player One
Score: 100
Move option queue: 1. ryott   2. chowkidar   3. cuirassier   4. fauj

Turn: Player One

Choose move option to [illegible] from queue (1 to 3) or 9 to take the offer
Enter the s[illegible] c[illegible]ning the piece to move (row number followed b
Enter the s[illegible] to move to (row number followed by column number):
New score:
```

```
   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

Turn: Player Two

Choose move option to [illegible] from queue (1 to 3) or 9 to take the offer
Enter the s[illegible] c[illegible]ning the piece to move (row number followed b
Enter the s[illegible] to move to (row number followed by column number):
New score:
```

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education



INSPECTION COPY

COPYRIGHT PROTECTED

INSPECTION COPY

Task 14

Coding:

- Using a method to track the Weather Event (this is the `weatherEvent` variable
- Having the countdown timer allow precisely two complete turns from when it is a
- Announcing to the players when the Weather Event start[illegible]ith a 2 turns warnin
- Destroying all pieces in the same column as the [illegible]ther Event when the timer
- Destroying a Kotla in the Weather Ev[illegible]n when the timer expires. [1 mar
- Correctly selecting a random [illegible]pty square. [1 mark]
- Creating a weather[illegible] class with `GetWeatherLocation` and `SetWeather`
- Implemen[illegible]un`CountDownComplete` so that it returns an appropriate value inclu expires. [1

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 f
                take the offer: "))
                if Choice == 9:
                    self.__UseTheCanOffer()
                    self.__DisplayState()
            while not SquareIsValid:
                StartSquareReference = self.__GetSquareReferenc
                move")
                SquareIsValid = self.__CheckSquareIsValid(Start
            SquareIsValid = False
            while not SquareIsValid:
                FinishSquareReference = self.__GetSquareReferen
                SquareIsValid = self.__CheckSquareIsValid(Finis
            #CODE ADDED
            if weatherEvent==None:
                weatherEvent = self.__WeatherEventOccurs()
            else:
                if weatherEvent.CountDownComplete():
                    ColToDestroy = weatherEvent.GetWeatherLocat
                    for Row in range (1,self._NoOfRows+1):
                        if self._Board[(self.__GetIndexOfSquare
                        GetPieceInSquare() != None:
                            self._Board[(self.__GetIndexOfSquar
                            Piece()
                        elif self._Board[(self.__GetIndexOfSquare
                        ContainsKotla():
                            self._Board[(self.__GetIndexOfSquar
                            Square()
            #END ADDITION
            MoveLegal = self._CurrentPlayer.CheckPlayerMove(Cho
            FinishSquareReference)
```

INSPECTION COPY
COPYRIGHT PROTECTED

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)
                random.randint(1,self._NoOfColumns)
                if self._Board[self.__GetIndexOfSquare(randomSqu
                == None and self._Board[self.__GetIndexOfSqu
                [illegible]Kotla():
                    squareFound = True
            weatherEvent=WeatherEvent(randomSquare)
        return weatherEvent
    #END ADDITION
```

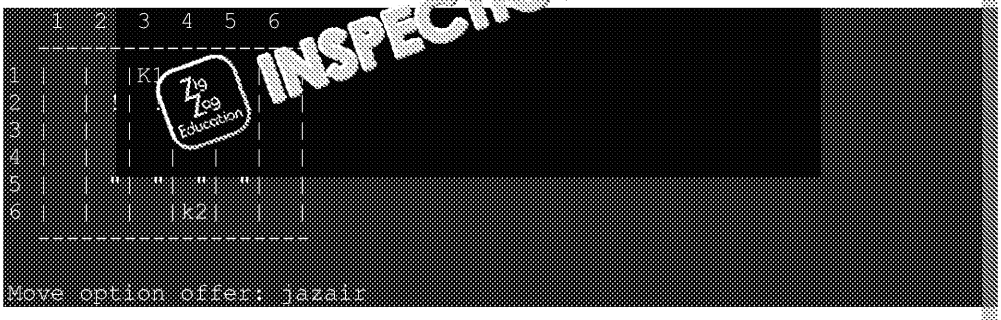
Code for WeatherEvent:

```
#CODE ADDED
class WeatherEvent():
    def __init__(self, SquareReference):
        self.__SquareReference = SquareReference
        self.__CountDownTimer = 3
        print("A weather event has occured at location",SquareRe
        print("After two complete turns, all pieces on the same

    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
            self.__SquareReference%10)
            return True
        else:
            self.__CountDownTimer -= 1
            print("The weather event at location", self.__Square
            "after one more turn" if self.__CountDownTimer>1 els
            return False
#END ADDITION
```

Testing:

- Having at least one piece owned by each player in the column where the weathe 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
```



INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

```
Player One
Score: 100
Move option queue: 1. ryott   2. chowkidar   3. cuirassier   4. fauj

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
A weather event has occured at location 61
After two complete turns, all pie[illegible] same column will be dest
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

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
The weather event at location 61 will occur [illegible] 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. ja

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
The weather event at location 61 will occur [illegible] turn
New score: 108

   1  2  3  4  5  6
 ------------------
1|  |  |K1|  |  |  |
2|  !|  |  |  |  !|
3|  |  |  |  |  |  |
4|  |  |  |  |  |  |
5| "|  | "| "| "|  |
6|  |  |  |k2|  |  |
 ------------------

Move option offer: jazair
```

INSPECTION COPY

COPYRIGHT PROTECTED

```
Player Two
Score: 104
Move option queue: 1. chowkidar   2. jazair   3. faujdar   4. cuiras

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
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

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
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 |  |  |  |  |  |  |
  -------------------

Move option offer: jazair

Player Two
Score: 108
Move option queue: 1. jazair   2. faujdar   3. cuirassier   4. ryott

Turn: Player Two

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

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Task 15

Coding:

- Creating a `Barrier` class that accepts the parameters `Player` and `Symbol` and [1 mark]
- Creating `ContainsBarrier` that returns true for B or b and false otherwise. [1
- Modifying `CheckSquareIsValid` to return false if the end square contains a ba
- Creating `CheckBarrierIsValid` which checks that all the squares for the barr
- Creating `CheckBarrierIsValid` which checks that all the squares for the barr
- Creating `PlaceBarrier` with suitable input messages that calls `CheckBarrier` creates a 3 squares wide on the board. [1 mark]
- Modifying `CreatePieces` to make two calls to `PlaceBarrier`, one for each pla
- Creating `CheckManhattanDistance` and modifying `PlayGame` to call that inste for the line starting `MoveLegal=`. [1 mark]
- Inside `CheckManhattanDistance`, checking that the start and end squares are `CheckPlayerMove`). [1 mark]
- Inside `CheckManhattanDistance`, iterating along the Row and Column and vi and has only been attempted. [1 mark]
- Inside `CheckManhattanDistance`, correctly iterating along the Row and Colum in all combinations of up, down, left and right (with and without vertical/horizonta possible move orientations (in the code below this was achieved by using the Ho `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.__CheckSquareIsInBounds(SquareReference):
            return
    #CODE
        if self.__Board[self.__GetIndexOfSquare(SquareReference)
            return False
    #END ADDITION
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

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
                self._Board[self.__GetIndexOfSquare(BarrierRow*10
                self._Board[self.__GetIndexOfSquare(BarrierRow*1
                != 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 l
            your horizontal, 3-square-wide barrier? "))
            if self.__CheckBarrierIsValid(BarrierSquare):
                BarrierValid = True
            else:
                print("You must choose the centre of 3 empty square
        BarrierRow = BarrierSquare // 10
        BarrierCol = BarrierSquare % 10
        for col in range(BarrierCol-1, BarrierCol+2):
            self._Board[self.__GetIndexOfSquare(BarrierRow*10+c
    #END ADDITION
```

Changes to C[redacted]P[redacted]ces (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 +
        CurrentPiece = Piece("mirza", self._Players[0], 5, "1")
        self._Board[self.__GetIndexOfSquare(10 + self._NoOfColumns
        #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
            .SetPiece(CurrentPiece)
        CurrentPiece = Piece("mirza", self._Players[1], 5, "2")
        self._Board[self.__GetIndexOfSquare(self._NoOfRows * 10
        + 1))].SetPiece(CurrentPiece)
        print("Player Two, it's time to place your barrier")
        self.__PlaceBarrier(self._Players[1],"b")
        #END CHANGE
```

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Changes to PlayGame (Dastan class):

```
                SquareIsValid = self.__CheckSquareIsValid(Finis

            #CODE CHANGE
            MoveLegal = self.__CheckManhattanDistance(Choice, St
            FinishSquareReference)
            #END CHANGE
            if MoveLegal:
```

Code for CheckManhattanDistance (Dastan class):

```
    #CODE ADDED
    def __CheckManhattanDistance(self, Choice, StartSquare, EndS
        if self._CurrentPlayer.CheckPlayerMove(Choice,StartSquar
            HorizontalDirection = -1 if EndSquare%10<StartSquare
            VerticalDirection = -1 if EndSquare//10<StartSquare/
            route1Valid = True
            for Row in range(StartSquare//10 + VerticalDirectio
            VerticalDirection, VerticalDirection):
                if self._Board[self.__GetIndexOfSquare(Row*10+St
                .ContainsBarrier():
                    route1Valid = False
            for Col in range(StartSquare%10, EndSquare%10, Hori
                if self._Board[self.__GetIndexOfSquare(EndSquare
                .ContainsBarrier():
                    route1Valid = False

            route2Valid = True
            for Col in range(StartSquare%10 + HorizontalDirectio
            HorizontalDirection, HorizontalDirection):
                if self._Board[self.__GetIndexOfSquare(StartSqua
                .ContainsBarrier():
                    route2Valid = False
            for Col in range(StartSquare//10, EndSquare//10, Ve
                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
- 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 of both routes and the t (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-s
Player Two, it's time to place your barrier
Which square would you like to be the centre of your horizontal, 3-s

   1  2  3  4  5  6
  -------------------
1 |  |  |K1|  |  |  |
2 |  | !| !| !| !|  |
3 |  |  |B |B |B |  |
4 |b |b |b |  |  |  |
5 |  | "| "| "| "|  |
```

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

```
6 |  |  |  |k2|  |  |
  -------------------

Move option offer: jazair

Player One
Score: 100
Move option queue: 1. ryott   2. chowkidar   3. cuirassier   4. fauj

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Choose the move option from your queue to replace (1 to 5): 1

   1  2  3  4  5
  -------------------
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. fau

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
New score: 96


   1  2  3  4  5  6
  -------------------
1 |  |  |K1|  |  |  |
2 |  | !|  |  | !|  |
3 |  |  |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

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):

   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
```

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

```
Move option queue: 1. chowkidar   2. cuirassier   3. faujdar   4. ja

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):

   1  2  3  4  5  6
 --------------------
1|  |  |K1|  |  |  |
2|  | !| !!|  | !|  |
3|  |  |B |B |B |  |
4|b |b |b |  |  | !|
5|  | "| "| "| "|  |
6|  |  |  |  |  |  |
 --------------------

Move option offer: ryott

Player Two
Score: 100
Move option queue: 1. ryott   2. chowkidar   3. jazair   4. faujdar

Turn: Player Two

Choose move option to use from queue (1 to 3) or 9 to take the offer
Enter the square containing the piece to move (row number followed b
Enter the square to move to (row number followed by column number):
Enter the square to move to (row number followed by column number):
New score: 104

   1  2  3  4  5  6
 --------------------
1|  |  |K1|  |  |  |
2|  | !| !!|  | !|  |
3|  |  |B |B |B |  |
4|b |b |b |  |  | !|
5| "|  | "| "| "|  |
6|  |  |  |  |  |  |
 --------------------

Move option offer: ryott

Player One
Score: 96
Move option queue: 1. chowkidar   2. cuirassier   3. faujdar   4. ja

Turn: Player One

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

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Name	

ZigZag Education supporting

A Level AQA Computer Science Pap

Summer 2023

Electronic Answer Document (EAD)

Instructions

- Enter your name in the box at the top of this page
- Answer **all** questions by entering your answers into this document
- Remember to **save** this document regularly
- Save and print this document and any additional pages
- Answer all questions
- The marks available for each question are shown in brackets
- You will need:
  - access to a computer
  - access to a printer
  - access to appropriate software
  - electronic copies of the required skeleton code
  - EAD (Electronic Answer Document)

Total marks:

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Exam-style Questions

Answer all questions. Remember to save this document

Q		Answer
1	(a)	
	(b)	
2	(a)	
	(b)	
3		
4	(a)	
	(b)	
5		
6	(a)	
	(b)	
7		
8	(a)	
	(b)	
9		
10	(a)	
	(b)	
11	(a)	
	(b)	
	(c)	
12		
13		
14	(a)	
	(b)	
	(c)	
	(d)	
15	(a)	
	(b)	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Exam-style Programming Task

Answer all questions. Remember to save this document

Q	Answe
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Preview of Questions Ends Here

This is a limited inspection copy. Sample of questions ends here to avoid students previewing questions before they are set. See contents page for details of the rest of the resource.

Preview of Answers Ends Here

This is a limited inspection copy. Sample of answers ends here to stop students looking up answers to their assessments. See contents page for details of the rest of the resource.