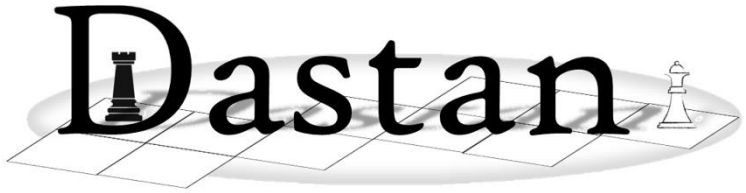


Exam-style Questions

These questions refer to the **Preliminary Material** and the **Skeleton Program**, but **do not** require any additional programming

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 the main game loop.

(a) State the time complexity of this loop. [1]

.....

(b) Explain the efficiency of this time complexity and how well it scales up. [3]

.....

.....

.....

.....

.....

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

Throughout the code there are many literals such as ‘mirza’, ‘jazair’, ‘ryott’, ‘faujdar’ and some others.

(a) Describe one problem that could occur due to this. [2]

.....

.....

.....

(b) Describe one possible solution to this problem. [2]

.....

.....

.....

3 This question refers to the private method **GetPointsForOccupancyByPlayer** in the **Dastan** class.

Explain precisely how polymorphism is used when calculating the **ScoreAdjustment** in this method. [3]

.....

.....

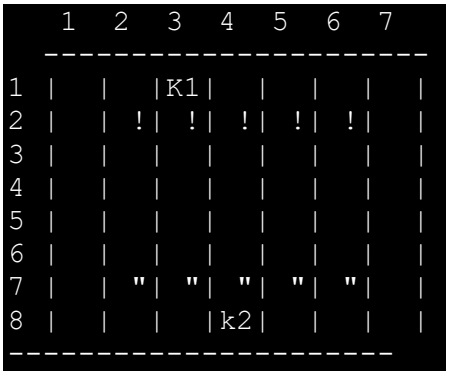
.....

.....

.....

4 This question refers to the **Main** method that is executed at the start of the game.

When **ThisGame** is instantiated, currently the arguments 6, 6, 4 are passed to **Dastan**.



(a) Assume the arguments 8, 7, 5 were passed to **Dastan** instead.

Explain why player one's Kotla and Mirza would appear in column 3 rather than in column 4 opposite player two's as per the image above. [2]

.....

.....

.....

(b) Describe how the code for the **CreateBoard** method of the **Dastan** class could be modified so that where there are an odd number of columns, then the Kotlas will both appear in the central column but when there are an even number it will remain as it is. [4]

.....

.....

.....

.....

.....

.....

5 This game refers to the private methods **CreateRyottMoveOption**, **CreateFaujdarMoveOption**,**CreateJazairMoveOption**, **CreateCuirassierMoveOption** and **CreateChowkidarMoveOption**.

Currently the methods take a **Direction** parameter which changes between 1 and -1 according to whose turn it is. Across the methods there is a lot of repeated use of the **Direction** parameter which always gets multiplied by any non-zero parameter to the constructor of **Move**.

Without suggesting any specific code, describe alternative logic that could remove the need for the **Direction** parameter by modifying the **AddToMoveOptionQueue** and **UpdateMoveOptionQueueWithOffer** methods of the **Player** class. [3]

.....

.....

.....

.....

.....

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 stack. [2]

.....

.....

.....

(b) Currently this class uses a list to store the queue data structure; explain how it could be modified to use an array to implement a circular queue with five elements.

You should not write any actual code for this question but refer to any new variables that may be required and create any algorithms using structured/descriptive English. Alternatively, you may produce an annotated diagram. [4]

.....

.....

.....

.....

.....

.....

.....

.....

7 This question refers to the method **GetIndexOfSquare** in the **Dastan** class.

Explain how the private method **GetIndexOfSquare** works. [3]

.....

.....

.....

.....

.....

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

(a) Explain how the board could be represented as a two-dimensional list or array. [2]

.....

.....

.....

(b) State one reason why an array is more appropriate to store the board than a list. [1]

.....

.....

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

Explain the purpose of metadata and give one example of metadata that could be stored for Dastan. [2]

.....

.....

.....

.....

10 This question refers to the **CreateMoveOptions**, **CreateMoveOption**, **CreateChowkidarMoveOption**, **CreateRyottMoveOption**, **CreateFaujdarMoveOption**, **CreateJazairMoveOption** and **CreateCuirassierMoveOption** methods in the **Dastan** class and to the **MoveOption** class.

(a) Currently the **MoveOption** class holds the details for whichever move is to be made that is generated/populated by one of the **CreateChowkidarMoveOption**, **CreateRyottMoveOption**, **CreateFaujdarMoveOption**, **CreateJazairMoveOption** and **CreateCuirassierMoveOption** methods in the **Dastan** class.

Explain why this is NOT polymorphism. [2]

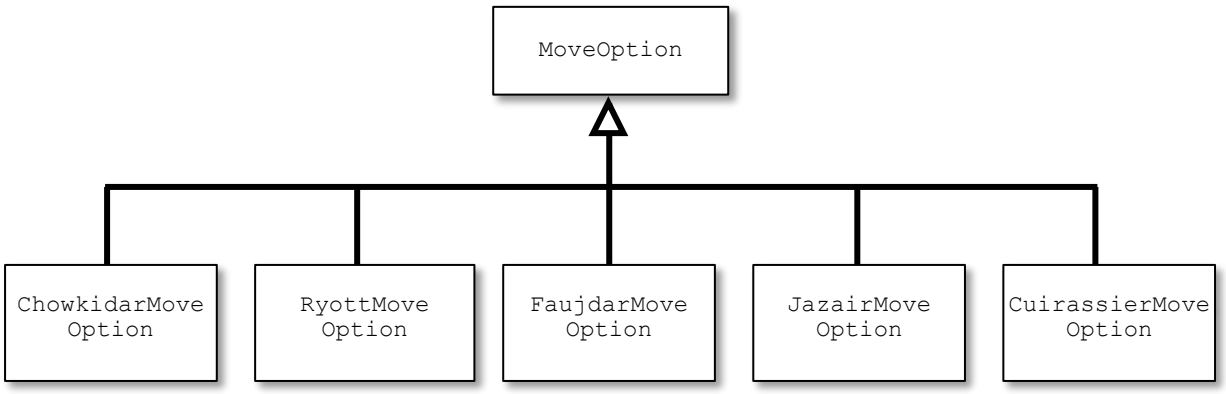
.....

.....

.....

.....

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



Explain how this inheritance structure could have been used effectively with polymorphism. [2]

.....

.....

.....

11 This question refers to the **Kotla** class.

(a) The constructor includes a call using **super();** explain the purpose of this. [2]

.....

.....

.....

(b) The method **GetPointsForOccupancy** has a different implementation from the method with the same name in the parent class. State the name for this OOP technique. [1]

.....

11 **(c)** Explain what the OOP technique *overloading* is used for. [3]

.....

.....

.....

.....

.....

12 The **MoveOptionQueue** class implements a normal queue, which is a FIFO (first in, first out) data structure.

Explain the different between a normal queue and a priority queue. [4]

.....

.....

.....

.....

.....

.....

13 This question refers to the constructor of the **Piece** class and the **SetPiece** method of the **Square** class.

Both methods take a parameter *P* which is unclear. Explain why variables should always have meaningful names. [2]

.....

.....

.....

14 This question is about access levels for attributes and methods and refers to the **Piece** class.

(a) The **Piece** class has four protected attributes; what does the word 'protected' mean in this context? [2]

.....

.....

.....

14 **(b)** The **Piece** class has four public methods; what does the word 'public' mean in this context? [1]

.....

.....

(c) There is one final level of access for attributes and methods which is private; what does that mean? [1]

.....

.....

(d) Why is it important to have access modifiers such as private, protected and public for both methods and attributes in OOP? [3]

.....

.....

.....

.....

.....

15 This question refers to the **CheckSquareInBounds** method of the **Dastan** class.

(a) This method uses integer division; explain the difference between integer division and floating point division. [2]

.....

.....

.....

(b) This method returns a Boolean value. Describe the meaning of Boolean. [1]

.....

.....

END OF QUESTIONS

