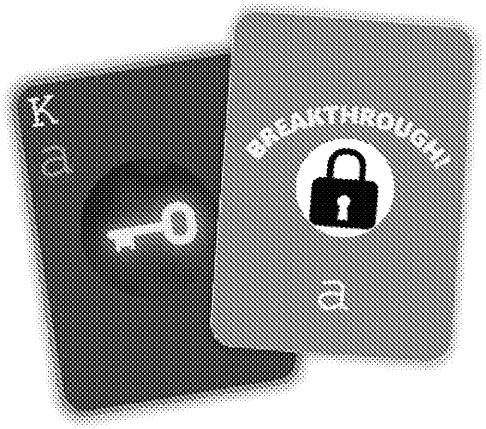
Zig Zag Education

Computer Science

A Level | AQA | 7517

2015 specification
for the 2022 exam

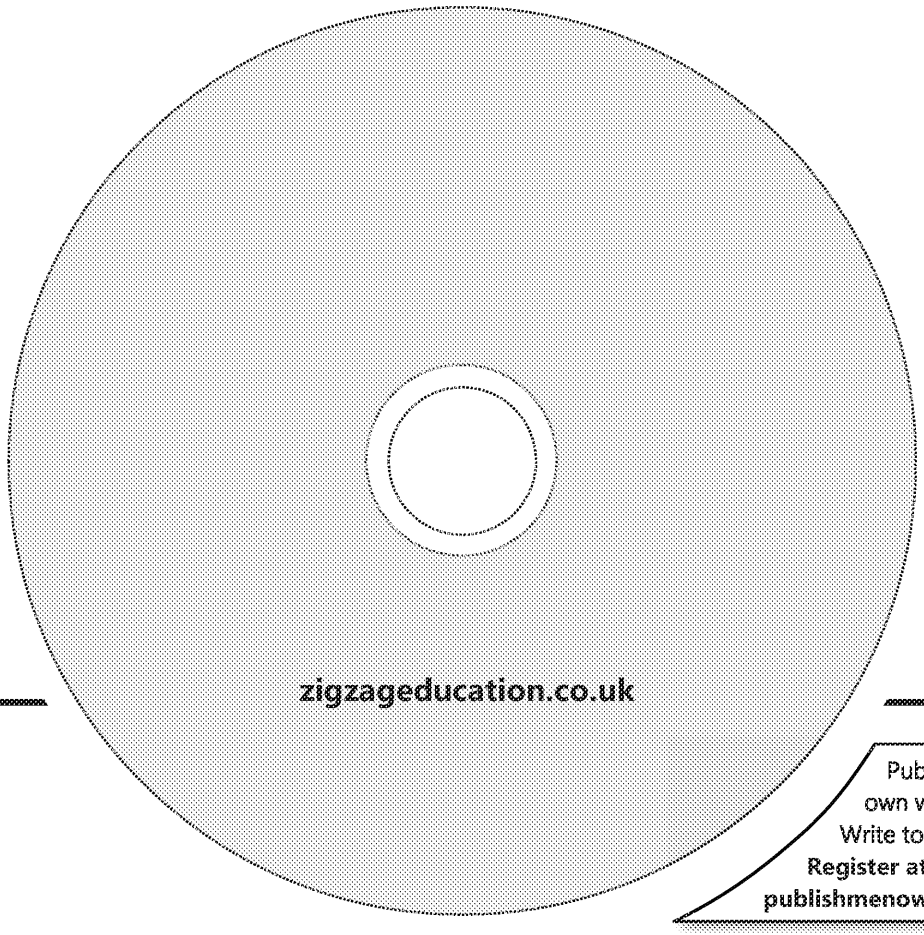


PAPER 1 EXAM RESOURCE PACK 2022

BREAKTHROUGH!

for A Level AQA Computer Science

PYTHON³ EDITION



DH8/
11149

POD
11149

zigzageducation.co.uk

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

Contents

** Note there are also electronic copies of the UML Diagrams ('Complete' & 'Activity' versions) on the CD – 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 ***Breakthrough!*** preliminary material (Python[3]) – for examination summer 2022.

On the CD, you will find the following:

`Breakthrough`	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: Copy the entire `Breakthrough` 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:

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**
Three UML Class Diagrams help students explore the skeleton program; there is a completed version, a partially-complete version (gap-fill), as well as a mostly blank template. The completed version is password-protected and accessible via the *Solutions* web page.

③ **Video**
Quick video going over the *Breakthrough!* card 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.**

BREAKTHROUGH

Skeleton Code Breakdow

Class: Breakthrough

Identifier / Data		Description
<<constructor>>		
Paramete		Initialises several private attributes inc • Deck to a new CardCollectio • Hand to a new CardCollectio • Sequence to a new CardColle • Discard to a new CardCollect • Score to 0 • GameOver to False • Locks to an empty list • CurrentLock to an empty Loc • LockSolved to False Invokes the LoadLocks() method to lo 'locks.txt'.
Return va	n/a	
AddDifficultyCardsToDeck (private)		
Parameters	n/a	Adds five Di ards to the Deck
Return values	n/a	
Check If Lo ha t (private)		
Paramete	n/a	Iterates through the Sequence CardC together the string SequenceAsString the separator between each card desc As a new element from Sequence is c SequenceAsString, the string is comp conditions using the CheckIfConditio lock to check whether a challenge has incrementally because challenges can challenge has been met, True is return returned.
Return values	Boolean	
CheckIfPlayerHasLost (private)		
Parameters	n/a	Checks to see if e are any cards le none, ar ate message is displ l score; the game is over a If there are cards still left in the Deck, False is returned, allowing the player t
Return values	Boolean	
CreateSt Deck (private)		
Parameters	n/a	Used by the SetupGame() method to the correct File, Pick and Keys for ea 5 Picks from toolkits a, b and c are ad Files and 3 Keys from toolkits a, b an
Return values	n/a	

Identifier / Data		Description
GetCardChoice (private)		
Parameters	n/a	Used by the PlayGame() method t their Hand they would like to use.
Return values	Value : Integer	Contains error handling to catch n not catch a out of range.
GetCardFromDeck (private)		
Parameters	CardChoice : Integer	Used to get the next card from the add it to the Hand.
Return values	n	If the Deck CardCollection has a system will then check if the card a a DifficultyCard. If a DifficultyCa they would like to lose a 'Key' card cards from the Deck. The Difficul Discard CardCollection and the on the DifficultyCard passing the parameters.
		The system then performs a check repopulating the Hand with cards If another Difficulty card is found Difficulty card (or cards if there is the Deck) is moved automatically CardCollection rather than into th
		If the D an out of cards, the
GetChoice (private)		
Parameters	n/a	Used by the PlayGame() method like to use a card from their Hand CardCollection on the screen.
Return values	: String	
GetDiscardFromPlayChoice (private)		
Parameters	n/a	Used by the PlayGame() method like to play the selected card from Discard the selected card from the CardCollection.
Return values	Choice : String	
GetRandomLock (private)		
Parameters	n/a	Returns a randomly selected lock Locks.
Return values	Lock	
LoadGame (private)		
Parameters	FileName : String	Use the FileName parameter to imports the current Score, Challe for the Hand, Sequence, Discard
Return values	Boolean	True is returned if the file is loade error occurs, an error message is returned.

Identifier / Data		Description
LoadLocks (private)		
Parameters	n/a	Uses a hard-coded 'lo locks available for the file contains the challe line from the file is spl Challenges, using a s Each Challenge is the as a delimiter into the The Conditions are th Lock variable – Locks to the private attribute If an error occurs, an e advise that the locks correctly.
Return values	n/a	
MoveCard (private)		
Parameters	FromCollection : CardCollection ToCollection : CardCollection CardNumber : Integer	Moves a card at the p the CardCollection F CardCollection ToC If the FromCollection ToCollection is the S been chosen (i.e. not score is updated appr ed. For all other m another, Score is not Score is returned.
Return values	Score : Integer	
PlayCardToSequence (private)		
Paramete	choice : Integer	This method is used t to the Sequence to te The system tests to s one card in the CardC system then checks t by the user is a differe played card. If the To can be played and the to the Sequence and appropriately for that then gets a new card Hand. If the Sequence does the system move Sequence and the S The system then uses CheckIfLockChallen the new card added t Challenge to be met appropriate message the player Score by 5
Return va	n/a	

Identifier / Data		Description
PlayGame (public)		
Parameters	n/a	This contains the main game loop.
Return values	n/a	Checks to confirm if the private list attribute Lock the LoadLocks() method. If none have been load screen and the program s.
		If the list does nt locks, it initialises the follow solved to False invokes the SetupGame() method to set
		The main game loop runs while the private attribu There is then an inner loop which runs while Gam attribute LockSolved is also False.
		The inner game loop displays the current user sc current lock and the contents of the Hand, and S
		Using the GetChoice() method to display a choic loop then uses selection to either display the Dis card in the game.
		If the user selects to use a card, the system uses to select a card. It then uses the GetDiscardOrP if the user wants to play or discard the chosen ca the system moves the selected card from the Ha CardCollection and gets a new card from the De GetCardFromDeck(). If the user selects play, the PlayCardToSequence() method to move the cho Sequence CardCollection.
		Once a card has been played or discarded, the m CheckLockSolved() method on the CurrentLock to challenges have been met. If they have, the Lock and a new lock is generated.
		If a lock has been solved, the inner loop returns b which checks if the game is over by invoking the method. If this returns True the game ends.
ProcessLockSolved (private)		
Parameters	n/a	Increments the Score by 10 and displays the use
Return values	n/a	Uses an indefinite loop to iterate through the Dis all of the cards back to the Deck.
		Reshuffles the Deck using the Shuffle() method the GetRandomLock() method with the private a

Identifier / Data		Description
SetupCardCollectionFromGameFile (private)		
Parameters	LineFromFile : String CardCol : CardCollection	Used for processing lines 4 to file which are for processing t CardCollections (namely the Sequence).
Return values	n/a	
		Receives a single line of text parameter) from the external processes it into a CardColle LineFromFile contains text, i SplitLine, using the comma a The SplitLine list is then proc card number and card type in CardCollection. If a Difficult instead of a normal ToolCard
SetupGame (private)		
Parameters	n/a	Called from the PlayGame() m message of the game on the s would like to load in an extern game. If the player chooses to system attempts to load the fi be loaded the game quits. If the player chooses to play a generates a new Deck using method and then shuffles it by method. It then moves 5 cards start the player off. The system AddDifficultyCardsToDeck() DifficultyCards into the Deck ensure they are in random loc assigns a new lock at random CurrentLock using GetRand
Return values	n/a	
SetupLock (private)		
Parameters	Line1 : String Line2 : String	Used for processing lines 2 a file which contain the challeng The parameter Line1 contains and the parameter Line2 cont Each line is split into a string delimiter. The Line1 parameter is then the delimiter to add a new cha single line may contain multip parameter is split using a sem populate the Met status for e SetChallengesMet() method
Return values	n/a	

Class: Challenge

Identifier / Data		Description
<<constructor>>		
Parameters	n/a	Initialises the following p • Met to False • Conditions to an
Return values	n/a	
GetCondition (public)		
Parameters	n/a	Returns a list of strings o challenge in the lock.
Return va	ition : List (String)	
GetMet (p		
Parameters	n/a	Returns the value of the
Return values	Met : Boolean	
SetCondition (public)		
Parameters	NewCondition : List (String)	Sets the value of the pro Condition from the para
Return values	n/a	
SetMet (public)		
Parameters	NewValue : Boolean	Sets the value of the pro parameter NewValue.
Return values	n/a	

Class: Lock

This class does not h ve a ecific constructor and the

Identifier / Data		Description
<<constructor>>		
Paramete	n/a	Initialises the Challenge empty list.
Return values	n/a	
AddChallenge (public)		
Parameters	Condition : List (String)	Initialises a new challeng condition from the para
Return values	n/a	
		Appends the new challe protected attribute.
CheckIfConditionMet (public)		
Parameters	Sequence : String	Returns True and sets th SetMet() if the Sequenc ch ge, otherwise it re
Return values	Boolean	
ConvertConditionToString (private)		
Parameters	C : List (Stri	Converts list of condition displaying on the screen parameter C, concatena ConditionAsString() us the delimiter.
Return values	C ssString : String	
GetChallengeMet (public)		
Parameters	Pos : Integer	Returns the Met status o Pos in the Challenges li
Return values	Boolean	

Identifier / Data		Description
GetLockDetails (public)		
Parameters	n/a	Used for displaying a challenge's c through the Challenges protected together the output string LockDe version of all the challenges for the been set so no
Return values	LockDetails: String	
GetLockSolved (public)		
Parameters	n/a	Returns the status showing if a loc through the Challenges protected there are any unmet ones, otherw
Return va	Boolean	
GetNumberOfChallenges (public)		
Parameters	n/a	Returns the number of Challenge number of challenges in this lock)
Return values	Integer	
SetChallengeMet (public)		
Parameters	Pos : Integer Value : Boolean	Uses the SetMet() method in the C attribute of a challenge at the posi list to Met or not Met using the Va
Return values	n/a	

Class: Card

Identifier / Data		Description
<<constructor>>		
Parameters	n/a	Initialises the CardNumber static attribute (class variabl increments the static attribut NextCardNumber which m and updated for all objects o Initialises the Score protect
Return values	n/a	
GetCardNumber (public)		
Parameters	n/a	Returns the value of the pro
Return values	CardNumber : Integer	
GetDescription (public)		
Parameters	n/a	Returns the protected attrib string.
Return values	CardNumber: String	
GetScore (public)		
Parameters	n/a	Retur e protected attrib
Return values	Score : Integer	
Process (public)		
Parameters	Deck : CardCollection Pile : CardCollection Hand : CardCollection Sequence : CardCollection CurrentLock : Lock Choice : String CardChoice : Integer	Base class method for the classes to override.
Return values	n/a	

Class: ToolCard (inherits from Card)

Identifier / Data		Description
<<constructor>>		
Parameters	args : List	Initialises the following protected attributes: • ToolType from parameter args[0] • Kit from parameter args[1] If there are three elements in args it will initialise a Kit from parameter args[2]; otherwise it will initialise Kit from parent constructor. Invokes the SetScore() method to assign the correct class for the ToolType.
Return values	n/a	
GetDescription (public)		
Parameters	n/a	Overrides the GetDescription() method from Card to return a concatenated string of the ToolType, a space,
Return values	String	
SetScore (public)		
Parameters	n/a	Assigns the correct Score from the protected
Return values	n/a	

Class: DifficultyCard (inherits from Card)

Identifier / Data		Description
<<constructor>>		
Parameters	args : List	Initialises the protected attributes
Return values	n/a	If there is one element in args it will initialise Kit from parameter args[1]; otherwise it will initialise CardNumber by calling the
GetDescription (public) <<override>>		
Parameters	n/a	Overrides the GetDescription() method from Card class to return the protected
Return values	String	
Process (public) <<override>>		
Parameters	Deck : CardCollection Discard : CardCollection Hand : CardCollection Sequence : CardCollection CurrentLock : Lock Choice : String CardChoice : Integer	Overrides the Process() method to process the user choices from the user receives a difficulty card and like to discard a key or 5 cards. On choosing the option to discard select a key. This method checks parameter is valid. ***Although there is in this check, AQA expect code is written as it was in***
Return values	n/a	
		If the Choice parameter contains converted to an index by subtracting a 'key' ToolCard in the player's from the Hand and placed in
		If the Choice parameter does through deliberate user choice removed from the Deck and CardCollection.

Class: CardCollection

Identifier / Data		Description
<<constructor>>		
Parameters	N : String	Initialises the following protected • Nam om parameter N is to an empty list
Return values	n/a	
GetCardDescriptionAt (public)		
Parameters	X : Inte	Returns a string containing the d X in the Cards list by invoking th method in Card.
Return va		
GetCardNumberAt (public)		
Parameters	X : Integer	Returns the CardNumber attribu the Cards list.
Return values	Integer	
GetName (public)		
Parameters	n/a	Returns the value of the protecte
Return values	Name : String	
AddCard (public)		
Parameters	C (Card)	Appends the value of parameter Cards.
Return values	n/a	
CreateLineOfDashes (private)		
Parameters	Size : Integer	Used in formatting a CardCollec
Return values	LineOfDashes : String	Returns an appropriately sized L of elements in a CardCollection CardCollection is greater than t Size).
GetCardDisplay (public)		
Parameters	n/a	Used in formatting a CardCollec display output of a CardCollecti the collection Name and card de list attribute Cards. If there are n collection name and 'empty' is re If there are cards in the collection which is either appropriately size the collection or is fixed at 10 if t collection is greater than 10. This fits correctly in the terminal windo It then uses indefinite iteration to using the GetDescription() meth of the card at each element and and the | (pipe) symbol to create It then creates a second line of d underneath the 'line of cards' and
Return values	CardDisplay : String	

Identifier / Data		Description
GetNumberOfCards (public)		
Parameters	n/a	Returns the number of cards in t Cards.
Return values	Integer	
RemoveCard (public)		
Parameters	CardNumber : Integer	Re tur he card from Cards list emoves it from Cards.
Return values	CardtoGet : Ca n	If CardNumber is not a valid inc uninitialised variable CardToGe
Shuffle (p		
Parameters	n/a	Uses definite iteration to perform from one random position to and attribute Cards in order to gener
Return values	n/a	

BREAKTHROUGH!

UML Class Diagram

BREAKTHROUGH

Theory Questions

These questions refer to the **Preliminary Material** and the **S**
but **do not** requi ny additional programmin

TOTAL MARKS: 80

1 Exami private method MoveCard. Currently this method return

(a) State a more appropriate name for this local variable.

.....

(b) Currently the MoveCard method returns an integer which represe that was moved. Sometimes this return value is ignored.

Evaluate the choice (of the programmer) to ignore the return valu return 0 in some cases, and suggest an alternative implementatio

.....

.....

.....

.....

.....

.....

.....

2 The class CardCollection currently contains an interface that expose structure of a list. For the sequence and the discard pile, a more appr would be either a queue or a stack.

(a) Justify whether you would use a queue or a stack. When giving y functionality of the data structure to the behaviour of the game.

.....

.....

.....

.....

.....

.....

.....

(b) In order to implement a stack or a queue for the sequence, justify (and thereof) that you would make to the inheritance structure.

.....

.....

.....

.....

(c) How would you create a new class to handle a CardCollection that in[illegible] encapsulation?

.....

.....

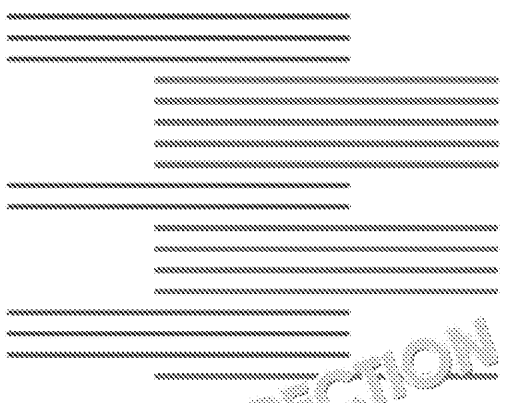
.....

.....

3 The Shuffle method of the CardCollection class currently swaps 10,[illegible] cards in order to shuffle the deck.

Another way of shuffling the deck is to use a method that humans would normally use called a 'riffle shuffle'. This involves splitting the deck into two approximately equal piles and then flicking through each pile from the top while combining the halves together into a single deck. Another way of thinking of this would be to imagine pushing the two halves together and having random number of cards between each card from each half as they recombine.

For example, a deck combined from a blue half and an orange half might look something like this:



Note that in the perfect case a riffle shuffle would use one card from each [illegible] is not desired, and in reality, between 0 and 5 cards will normally interleave [illegible] other half at any time.

a) Write a detailed algorithm for riffle shuffle in any format you choos
pseudocode, flow chart).

b) Explain the space complexity of your algorithm.

.....

.....

4 Examine the **CheckIfLockChallengeMet** method of the **Breakthroug**
CheckIfConditionMet method of the Lock class.

```
Lock:
Challenge Met: P c, F c, K c
Not met: P a, F a, P a
Sequence: P c, F c, K c, P a, F a, P a
```

(a) For the above sequence and lock, complete the trace table below
CheckIfLockChallengeMet method of the **Breakthrough** class.

Count	SequenceAsString	Return val
	""	
5		

(b) If the above lock had a third challenge as below, then how would (please fill it in below)?

```
Not met: F a, P a
```

Count	SequenceAsString	Return value
	""	
5		

5 Examine the **ProcessLockSolved** method in the **Breakthrough** class methods called by that method.

(a) When a new lock is set, if that lock has been solved before, it will automatically replaced with a new lock the following turn (and trea just solved the first new lock) but reward the player for solving the

.....

.....

.....

.....

(b) D e logical change you would make to the code (no nee al you can) to ensure that this no longer happens.

.....

.....

.....

.....

6 Examine the **Shuffle** method in **CardCollection**. This method will ma of cards in the deck.

(a) Explain how the effectiveness and eff of this algorithm dec cards in the deck reduces.

.....

.....

.....

.....

.....

(b) Other than introducing a riffle shuffle, justify how you could impro efficiency of the algorithm by describing any changes below.

.....

.....

.....

.....

7 ToolC ca be instantiated with either two or three arguments.

(a) Explain what happens in the case where a third argument is supp where only two arguments are supplied.

.....

.....

.....

.....

.....

(b) State the purpose of a constructor.

.....

.....

8 Exami e classes Card, ToolCard and DifficultyCard.

(a) Using evidence from these classes in the program, explain the dif abstract and a concrete class.

.....

.....

.....

.....

.....

.....

.....

(b) Using evidence from the Card method, explain the difference bet
(static) and an attribute.

.....

.....

.....

.....

9 Find a m e code for each of the following. Only write out th
line/s e.

(a) Inheritance

.....

.....

(b) Aggregation association

.....

.....

(c) A dynamic data structure

.....

.....

10 This q n refers to the concept of polymorphism and how it is use

(a) Choose and then write out one or more lines of the skeleton prog
polymorphism and justify why this is an example of polymorphism

.....

.....

.....

.....

.....

.....

(b) Define the term ' orphism'.

.....

.....

.....

.....

11 A suggestion has been made to introduce a new AdvancedLock that challenge which is only revealed once the basic challenges have been

Explain the steps that you would take in order to do this, i.e. the logical change/addition and the reason for each step.

You are not required to implement this or to write any actual code.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

12 Examine the Process method in the DifficultyCard class and the PlayCardToSequence and GetCardFromDeck methods in the Breakthrough class.

Using the sequence below:

```
Not matched: P a, F a, K a
Sequence: P a, F a
Hand: P b, K a, F b, K c, P a
```

The player plays the 'K a' card to the sequence and then draws a difficulty card which require them to either discard a key or five cards from the deck. The player must discard the 'K c' from their hand, which is currently in position 4.

Explain what will happen when the Process method is called under these circumstances, including specific references to the lines of code executed and in which order, and the values of variables, especially ChoiceAsInteger.

You will need to ensure that you look at the PlayCardToSequence and GetCardFromDeck methods in Breakthrough to be certain of the state of the Hand and Sequence when the DifficultyCard is drawn.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

13 The terms 'HAND', 'SEQUENCE', 'DECK' and 'DISCARD' all appear a
and in some cases more than once. This is an example of hard-coded
difficult to maintain and understand and also make it more prone to er

(a) Describe one method of avoiding hard-coding values that makes

.....

.....

.....

.....

(b) Explain why using hard-coded values makes the code more pron
understand.

.....

.....

.....

.....

14 Exception handling is used in several places in the skeleton code; two
the use of file handling.

(a) Describe why it is important to always use exception handling wh

.....

.....

.....

.....

(b) G example of another situation (not file handling) where exc
(it doesn't have to be from the skeleton code) and explain why.

.....

.....

.....

.....

15 This question refers to the PlayGame method of the Breakthrough c

Explain the use of the private attribute GameOver in this method, spe
is set and why it is used as the condition for iterative statements.

.....

.....

.....

.....

END OF QUESTIONS

BREAKTHROUGH

Theory Questions

These questions refer to the **Preliminary Material** and the **S**
but **do not** requi[illegible] ny [illegible]dditional programmin

TOTAL MARKS: 80

1 Exami[illegible] private method MoveCard. Currently this method return

(a) State a more appropriate name for this local variable.

(b) Currently the MoveCard method returns an integer which represe
that was moved. Sometimes this return value is ignored.

Evaluate the choice (of the programmer) to ignore the return valu
return 0 in some cases, and suggest an alternative implementatio

2 The class CardCollection currently contains an interface that expose
structure of a list. For the sequence and the discard pile, a more appr
would be either a queue or a stack.

(a) Justify whether you would use a queue or a [illegible]k. When giving y
functionality of the data structure to t[illegible] el[illegible]iour of the game.

(b) In order to implement a [illegible] queue for the sequence, justify
thereof) that you [illegible]ke to the inheritance structure.

(c) H[illegible]ating a new class to handle a CardCollection that
im[illegible]encapsulation?

3 The Shuffle method of the CardCollection class currently swaps 10,0
cards in order to shuffle the deck.

Another way of shuffling the deck is to use a method that humans would normally use called a 'riffle shuffle'. This involves splitting the deck into two approximately even piles and then flicking through each pile from the bottom while combining the halves together into a single deck. Another way of thinking of this would be to imagine pushing the two halves together and having a random number of cards between each card from each half as they r[illegible]mbine.

For example, a deck combined from a bl[illegible] and an orange half might look somethi[illegible] is:

Note that in the p[illegible] a riffle shuffle would use one card from e
is not [illegible]d, [illegible] reality, between 0 and 5 cards will normally inter
other [illegible] any time.

a) Write a detailed algorithm for riffle shuffle in any format you choos
pseudocode, flow chart).

b) Explain the space complexity of your algorithm.

4 Examine the **CheckIfLockChallengeMet** method of the **Breakthrough** **CheckIfConditionMet** method of the **Lock** class.

```
Lock:
Challenge Met: P c, F c, K c
Not met: P a, F a, P a
Sequence: P c, F c, K c, P a, F a, P a
```

(a) For the above sequence and lock, complete a trace table like the **CheckIfLockChallengeMet** method of the **Breakthrough** class.

Count	SequenceAsString	Return value
	""	
5		

(b) If the above lock had a third challenge as below, then how would (Complete an updated trace table)

```
Not met: F a, P a
```

Count	SequenceAsString	Return value
	""	
5		

5 Examine the **ProcessLockSolved** method in the **Breakthrough** class methods called by that method.

(a) When a new lock is set, if that lock has been solved before, it will automatically replaced with a new lock the following turn (and trea just solved the first new lock) but reward the player for solving the

(b) Describe the logical change you would make to the code (no nee although you can) to ensure that this no lon appens.

6 Examine the **Shuffle** method in **CardCollection**. This method will mak of cards in the deck.

(a) Explain how the effectiveness and efficiency of this algorithm dec cards in the deck reduces.

(b) Other than introducing a riffle shuffle, justify how you could impro efficiency of the algorithm by describing any changes below.

7 ToolCards can be instantiated with either two or three arguments.

(a) Explain what happens in the case where a third argument is supp where only two arguments are supplied.

(b) State the purpose of a constructor.

8 Examine the classes Card, ToolCard and DifficultyCard.

(a) Using evidence from these classes in the program, explain the dif abstract and a concrete class.

(b) Using evidence from the Card method, explain the difference bet (s and an attribute.

9 Find an example in the code for each of the following. Only write out th line/s of code.

(a) Inheritance

(b) Aggregation association

(c) A dynamic data structure

10 This question refers to the concept of polymorphism and how it is use

(a) Choose and then write out one or more lines of the skeleton prog polymorphism and justify why this is an exam of polymorphism

(b) Define the term 'polymorphism'.

11 A suggestion has been made to introduce a new AdvancedLock that challe hic only revealed once the basic challenges have been

Explain steps that you would take in order to do this, i.e. the logica change/addition and the reason for each step.

You are not required to implement this or to write any actual code.

12 Examine the Process method in the DifficultyCard class and the Pla GetCardFromDeck methods of the Breakthrough class.

Using the scenario below:

```
Not met: P a, F a, K a
Sequence: P a, F a
Hand: P b, K a, F b, K c, P a
```

The player plays the 'K a' card to the sequence and then draws a diffi require them to either discard or five cards from the deck. The p discard the 'K c' from hand, which is currently in position 4.

Explain what happen when the Process method is called under th includi specific references to the lines of code executed and in whic values of variables, especially ChoiceAsInteger.

You will need to ensure that you look at the PlayCardToSequence an methods in Breakthrough to be certain of the state of the Hand and S the DifficultyCard is drawn.

13 The terms 'HAND', 'SEQUENCE', 'DECK' and 'DISCARD' all appear a
and in some cases more than once. This is an example of hard-code
difficult to maintain and understand and also make it more prone to er

(a) Describe one method of avoiding hard-coding values that makes

(b) Explain why using hard-coded values makes the code more pron
understand.

14 Exception handling is u in several places in the skeleton code; two
the use of file ha

(a) D why it is important to always use exception handling wh

(b) Give an example of another situation (not file handling) where ex
(it doesn't have to be from the skeleton code) and explain why.

15 This question refers to the PlayGame method of the Breakthrough c

Explain the use of the private attribute GameOver in this method, spe
is set and why it is used as the condition for two iterative statements.

END OF QUESTIONS

BREAKTHROUGH

Programming Tasks

These questions require you to load the **Skeleton Program** and to make

Note that any alternative or additional code changes that you deemed appropriate – ensuring that it is clear where in the Skeleton Program those changes

Task 1

This question refers to the PlayGame method of the Breakthrough class.

The number of cards left in the deck should be printed out after the current cards in the player's hand each turn.

Test the changes you have made:

Run the game and play two turns, showing the number of cards in the deck

Evidence that you need to provide:

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

Task 2 D

This question refers to the PlayGame and GetChoice methods of the Bre
creation of a new attribute (with accessor methods), PeekUsed in the Loc

Introduce a **(P)eek** option. This can be used once per lock, and allows a p
see the next three upcoming cards. There should be a new command in P
the 'deck peek' is still available.

Create a new attribute in the Lock class called PeekUsed. Create access
to update and read PeekUsed attribute (get/set).

Update the GetChoice() method in the Breakthrough class to give the us
menu option should only appear if the PeekUsed attribute is False.

Introduce an option to the menu in the PlayGame() method to accept 'P' a
This menu option should only appear if the PeekUsed attribute is False. D
the deck using the GetCardDescriptionAt() method. Set the PeekUsed a
peek option has been chosen by the user.

When the player is given a new lock, set the PeekUsed attribute appropria
the peek option again.

Test the changes you have made:

Run the game and peek (peek is an option, it works) and then it's no longe
make sure it doesn't work even though the option isn't displayed. Solve a l
now an option again.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE showing changes made to the GetCho
- PROGRAM SOURCE CODE for the new PeekUsed attribute
- SCREEN CAPTURE(S) showing the required test.

Task 3

This question refers to the PlayCardToSequence method of the Breakth

Under the rules of the game, a player cannot play two cards of the same t there is no error message warning the player when they attempt to do this

Modify the PlayCardToSequence method in the Breakthrough class to i which tells the user that they cannot play two cards of the same type sequ

Use the GetCardDescriptionAt method to highlight to the user which card and explain why it is the same as the type just played.

Test the changes you have made:

Run the game and show at least one turn played where the error does not shows the new error message under the correct conditions of playing a du show that (1) the error message is displayed and (2) the card is not played

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayCa
- SCREEN CAPTURE(S) showing the required t

Task 4

This question refers to the PlayGame and GetChoice methods and the cr
attribute, MulliganUsed of the Breakthrough class.

Each player gets 1 'mulligan' per game where they can take all the cards i
discard pile and the sequence, put them together a shuffle up and deal
drawn (when repopulating the player's hand) should be sent to the discard
the current lock including any solved challenges will remain unchanged.

Create a new attribute for the Breakthrough class called MulliganUsed w
the Mulligan Used is False then display an additional **(M)ulligan** option ea
once the mulligan has been used, set the MulliganUsed attribute to True,
(M)ulligan option is no longer displayed or usable.

Test the changes you have made:

Run the game, solve one challenge, use mulligan, play one card to the se
attempt to mulligan again despite no menu option).

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE showing changes made to the Breakt
- PROGRAM SOURCE CODE showing changes made to the GetCho
- SCREENSHOT(S) showing the required test

Task 5 D

This question refers to the PlayGame and GetChoice methods of the Bre

The player will have a new option in PlayGame to **(Q)uit**, and for this they score for each card remaining in the deck. Print out their final score as the

Note that the code should exit cleanly/nicely without using any system.ex statements, although break/continue are allowed of course.

Test the c s you have made:

Play one turn of a game, choose quit.

Evidence that you need to provide:

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

Task 6 Diffi

This question refers to the GetCardFromDeck method of the Breakthrou
new method, DisplayStats, modifying two existing methods, AddCard an
adding three new attributes, NumPicks, NumFiles and NumKeys, in the

Introduce a stats / card count to the CardCollection class which keeps tra
out of the deck and calculates the % chance that the next card tile in the d

Introduce three new attributes to the CardCollection class called NumPic
which will be updated every time a ToolCard is added to or removed from

Create a new method in the CardCollection class called DisplayStats. T
the percentage chance of the next card being a key, pick or file based on t
the number of cards left in the deck.

When the player receives a difficulty card, use the DisplayStats method to
GetNumberOfCards method in the CardCollection class to display the fo
they choose 'lose a key or discard 5 cards from the deck'.

*There is a X% chance that the next card will be a key, a Y% chance that it
that it will be a pick.*

The percentages should be displayed to two decimal places.

Replace X, Y and Z with the appropriate values. Note that they will not nor
because there are also difficulty cards in the deck.

Test the changes you have made:

Run the game until a difficulty card is drawn and show the printout of the s
(after the hand and before asking which card).

Evidence that you need to provide:

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

Task 7 Diffi

This question involves the CreateStandardDeck, ProcessLockSolved a methods of the Breakthrough class, as well as the creation of a new SetC ToolCard and CardCollection classes.

Introduce three new 'multi-tool' cards – a **multi-** **), a multi-key (K**

At the start of a standard game (n ading a save game file), the d of these new types of car ti-tool cards can be dealt to the player's ha cards are.

On playing ti-tool card, the player should be given the option to choo assign the card to before it is added to the sequence, therefore allowing a any lock challenge of that type.

When a lock has been solved, three new multi-tool cards (one of each typ available for the next lock and the deck is reshuffled (as normal).

Test the changes you have made:

Play the game and show the use of at least one multi-tool card, the print s sequence both before and after the multi-tool is played.

Evidence that you need to provide:

- PROGRAM SOURCE CODE sh changes made to the CreateS
- PROGRAM SOU showing changes made to the Process
- PRO SOURCE CODE showing changes made to the PlayCar
- PROGRAM SOURCE CODE for the new SetCardToolkit method (in CardCollection classes)
- SCREEN CAPTURE(S) showing the required test.

Task 8 Diffi

This question refers to the GetLockDetails method of the Lock class and Breakthrough class.

Challenges are to be marked as 'partially met' (rather than just 'met' or 'no solved. A challenge is partially met if the end of the sequence (last one or an unsolved challenge.

Modify the call to GetLockDetails from PlayGame to pass in the sequenc

Modify GetLockDetails so that if the challenge is not met then it checks to For challenges of three cards, only check the last two cards and it become of the sequence matches the first card of the challenge or the second last the first card of the challenge and the last card of the sequence matches t challenge.

In general, check N-1 cards where N is the number of cards in the challen challenges of one card cannot be partially met. You only need to solve the three cards exactly.

Test the changes you have made:

Run the game and play one card to the sequence that doesn't match any one towards one of the three card challenges that matches the first card fo screen showing this entire turn.

Then play a second card to sequence that matches the second card o

Evidence that you need to provide:

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

Task 9 Diffi

This question refers to the PlayGame method of the Breakthrough class.

Introduce a bonus for solving locks using fewer cards. Once the first card is sequence for a new lock, a counter starts and one is added every time a pl (discarding or playing to the sequence).

Once a lock is solved (all the chains are), a player receives an extra point f the counter, after which it is reset. The player simply receives 0 if the count message c[illegible] bonus points that were awarded (including 0 if that

Test the changes you have made:

Run the game and play two locks, one solved in under 20 cards to show a in over 20 cards to show a bonus score of 0.

Evidence that you need to provide:

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

Task 10 Diffi

This question refers to the ProcessLockSolved, SetupGame and GetCa as the creation of a new method, AddGeniusCardToDeck of the Breakth of a new class called GeniusCard.

Introduce a new 'Genius Card' which is added to de at the start of a lock chance of having a 'Genius Card' in a de

A player can choose to use the 'Genius Card' when they draw it to solve a ask which card will be discarded and then reshuffled into the de from the discard pile.

Note that if a GeniusCard is drawn when filling up the hand it should be d message should be printed to this effect.

Create a method called AddGeniusCardToDeck which has a 25% chanc to the deck. This should be called from ProcessLockSolved and SetupG

Create a new class for the GeniusCard which inherits Card with CardTyp the GetCardFromDeck method of Breakthough to ensure that the card is drawn.

Test the changes you have made:

Run the game and play until a 'Genius Card' is drawn, then choose yes an challenge in the current lo

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the Proces
- PROGRAM SOURCE CODE showing changes made to the SetupG
- PROGRAM SOURCE CODE showing changes made to the GetCar
- PROGRAM SOURCE CODE for the new GeniusCard class
- PROGRAM SOURCE CODE for the new AddGeniusCard method
- SCREEN CAPTURE(S) showing the required tes

Task 11 Di

This question refers to the addition of a new attribute in the Breakthrough cl
GetCardFromDeck method of the Breakthrough class as well as the creat
PrintToolsAvailable, for the CardCollection class.

Introduce the concept of 'Buying a tool' from the dec

Add a new attribute, Credits, to the Br ough class which contains the
currently has. At the start of the gan s player has 10 credits. When a play
sequence or discarded c t they have at least 2 credits remaining, they s
like to buy (y/ ore their hand is refilled from the deck. If they choos
card as no t otherwise the new card will be the tool card that they purc

Players can 'buy' a 'Key' card at the cost of 3 credits, and 'file' or 'pick' cards a
When the player chooses 'y' to buy a tool, they should be prompted with the f
have 0 availability should not be listed).

```
1.    F a (1 available)
2.    F b (1 available)
3.    F c (1 available)
4.    P a (1 available)
5.    P b (1 available)
6.    P c (1 available)
7.    K a (1 available)
8.    K b (1 available)
9.    K c (1 available)
10.   No Tool (buy nothing)
```

***Note**: the actual number a*

***Note**: keys (items 7–9) sho*
at least 3 credits left. All m
numbers given above even
e.g. item 10 should always
the player changed their m

The new PrintToolsAvailable meth s d take one parameter, KeysAv
player has at least 3 credits e otherwise is False. It should return an array
available tool card a ype, for example, if the deck contains three files fr
at index 3 deck, no files from toolkit b and one file from toolkit c which is
the first par e array returned would look like this:

```
[ 3, -1, 12, ...
```

***Note:** -1 is used to indicate*

Test the changes you have made:

1. Run the game and play any card to the sequence, then choose 'y' whe
a tool. Select any tool listed as available, play it to the sequence and t
asked if you would like to buy a tool; show all the output produced incl
and the tool card being added to the player's hand each time.
2. Continue playing the game and buying tools until you have spent a tota
pick/file and 2 keys) and then show the printed li tools available wh

Evidence that you need to p

- PROGRAM SOU CODE showing changes made to the GetCardF
- PRO SOURCE CODE for the new Credits attribute
- PROGRAM SOURCE CODE for the new PrintToolsAvailable method
- SCREEN CAPTURE(S) showing the required test

Task 12

This question refers to CheckIfLockChallengeMet method of the Breakth

Create an 'Advanced' mode where, for any challenge that requires three c once the challenge is solved move the cards used to solve it from the seq exposing the previous card on the sequence, which could then possibly be challenge.

For example, if the sequence contains:

```
Fa, Kc,
```

and the current challenge is Pb, Kb, Fb. Suppose you play Kb and Fb to th current challenge but instead of the sequence extending to:

```
Fa, Kc, Pb, Kb, Fb
```

it will be contracted to:

```
Fa, Kc
```

and the Pb, Kb and Fb cards from the challenge that was just solved will b

Test the changes you have made:

Run the game and restart until you get a Lock and at least one challenge o of three cards. Play until you solve the three card challenge and then play challenge. The screen capture(s) should show the Lock, Sequence and Ha card to solve the three card challenge and the Lock and Sequence after yo

Evidence that you need to provide:

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

Task 13

This question refers to PlayGame and GetChoice methods and to the cre
method in the Breakthrough class. It also requires the creation of new Ge
GetChallengesMetAsString methods in the Lock class.

The PlayGame menu should have a **(S)** ve option which will save the gam
allow it to be reloaded (from the main menu when you first start the game).

In order to understand the format of the save game file, you will have to in
LoadGame method of the Breakthrough class.

Print out a suitable message stating whether the game was saved succes

Test the changes you have made:

1. Take a copy of the **game1.txt** file and rename it **backup.txt**.
2. Run the game until you get a lock with at least two challenges. Sol
save the game as '**game1.txt**' (it shouldn't prompt you). Load the g
of it has been correctly restored.
3. Restore the original **game1.txt** from **backup.txt**.

Evidence that you need to

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE showing changes made to the GetCho
- PROGRAM SOURCE CODE for the new SaveGame method
- PROGRAM SOURCE CODE for the new GetChallengesAsString m
- PROGRAM SOURCE CODE for the new GetChallengesMetAsStri
- SCREEN CAPTURE(S) showing the required test

Task 14

This question refers to PlayGame and PlayCardToSequence methods an
attribute, BonusPool, in the Breakthrough class. It also requires the crea
IsPartial, in the Lock class, which takes Sequence as a parameter.

Introduce a bonus for playing consecutive cards towards a challenge that solves a cha
card played in a row that goes towards solving a challenge will add 5 to th
pool will be added to the score for the card.

For example, if the bonus pool is 0 and a player plays a card towards challen
added to the score along with their normal score and the bonus pool is in
does anything except play another correct card towards challenge 1, then
0; otherwise they will get the score for the card played as normal, plus the
bonus pool will be increased to 10 and so on.

Test the changes you have made:

Run the game and keep discarding until you have all three cards required
solve it one card after another; continue playing and play a card to a challe
sequence that is not part of the challenge.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the PlayGa
- PROGRAM SOURCE CODE showing changes made to the PlayCa
- PROGRAM SOURCE CODE for the new BonusPool attribute
- PROGRAM SOURCE CODE for the new IsPartial method
- SCREEN CAPTURE(S) showing the required test

Task 15 Diffic

This question refers to the ProcessLockSolved, GetCardFromDeck and
methods, and to the creation of new private GenerateSolubleLock and G
and a new private attribute FinalLock in the Breakthrough class. It also r
public IsSoluble method in the Lock class that takes in Deck and Hand

EXTRA FILE NEEDED: **game2.txt**

Every lock generated must be solvable based on the cards left in the deck
exhaust the deck. If the lock cannot be solved, then choose a new random
in a row (without a suitable lock being found) then display a message 'Fina
generate a lock with two challenges that can be solved.

Once those challenges are solved, there should be a message from Check
instead of saying the player lost, prints out 'You have solved the final lock.

When approaching this task you should ignore the effect of Difficulty card
check that the Deck and Hand combined contain the requisite number of
lock.

The attribute FinalLock should be set to 0 at the start and then set to 1 in
the final lock is set. When CheckIfPlayerHasLost runs it should set Fina
that the final turn is played). If FinalLock == 1 and there are no cards left
doesn't lose until all the cards from the hand are gone.

Test the changes you have made:

1. Change the game to load the file **game2.txt** instead of **game1.txt** a
load game.
2. Play the game until the message 'Final Lock' is displayed, then sol
final turn.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the ProcessL
- PROGRAM SOURCE CODE showing changes made to the CheckIfP
- PROGRAM SOURCE CODE showing changes made to the GetCardF
- PROGRAM SOURCE CODE for the new GenerateSolubleLock meth
- PROGRAM SOURCE CODE for the new GenerateChallenge method
- PROGRAM SOURCE CODE for the new IsSoluable method
- PROGRAM SOURCE CODE for the new FinalLock attribute
- SCREEN CAPTURE(S) showing the required test

BREAKTHROUGH

Possible Additional Coding T

1. Create an extra toolkit (e.g. 'd') and ad[illegible] involving this to the l
2. Introduce a *Swiss Army K*[illegible] which can be used as any singl toolkit.
3. Add [illegible] ful[illegible]ity to allow a player, at the start of a turn, to (U)n rev[illegible]rything – including the score, sequence, hand and discar the start of the previous turn. There should be one undo available p possible to use it on the first turn of a new lock.
4. Add a *High Scores* file and ability to view this from a main menu.
5. Add levels so that different locks have different challenges which w depending on the current level. This could be linked to (11 – comp number of toolkits used, e.g. 2, 3 or 4.
6. Add a *Mighty Hammer* card that can smash (solve) the current lock your hand and play it later.
7. Introduce a user-defined locks option. This generates a rough pseu one player can choose a lock sequence and another has to try to u original game of *Mastermind*). A user-defined lo[illegible] must follow the at least two must be files, and at least on[illegible] a pick.
8. Introduce a second type of lo[illegible] Maths Lock', whereby the play the way they are now [illegible] new maths locks. This will involve each card call[illegible]ber'. The value of 'number' is displayed in a (pi[illegible] currently displayed. Cards can be used for their ma the[illegible] attribute. For example, if a lock contains four files – each that gives a total lock value of 20. The player needs to play a sequ a total value of 20. For example, if the player plays two picks, each then the lock will open. These new 'Maths Locks' are solved only u are independent of the tool type and tool kit. Receive a bonus of 5 challenge could never be solved (with the cards and deck as they a
9. Add an *Autoplay* mode which shows a computer simulation of the g
10. Design a formula to compute a complexity value for a lock.
11. Validation of card to play (with exception handling) for choosing wh response to a difficulty card.
12. Validation on entry of choice (or any entry) s[illegible] the player can c
13. Be able to sacrifice a card (remo[illegible] game) in order to cha
14. Examine the game1.t[illegible] game) closely and draw a flow diag player could [illegible]ove from the 'saved state' to 'end of the ga loc[illegible]t t[illegible]ata in the game1.txt file rather than playing the gam

BREAKTHROUGH!

Theory Questions (Mark Scheme)

Question		Suggested Solution	Total Marks	Marking Guidance
1	(a)	e.g. AdditionalScore // CardScore	1 mark	**A:** Similar names with meaning to explain the score. **R:** Spaces in names. **I:** case.
	(b)	**4 marks:** Ignoring a return value is not good practice **[1]**… One alternative would be to create a new method called *MoveCardWithScore* **[1]**… which takes in current player *Score* as a parameter and returns the updated *Score* **[1]**… and to remove the return value from the current method **[1]**. **Examples of answers worth less than full marks:** **3 marks:** make the return type void **[1]**… and move the logic to the place where the card is played to the sequence **[1]**… which is the only time the score is needed **[1]**. **2 marks:** remove the scoring **[1]**… and create a separate *getScore()* method **[1]**. **1 mark:** always check the return value as ignoring it is bad practice **[1]**.	4 marks	**A:** any reasonable suggestion. **A:** answers without passing *Score* as a parameter and dealing with the extra score as per now. **A:** answers where there is a scoring method in *CardCollection* which 'knows' whether to score or not. **A:** passing score in by reference and having a new attribute on *CardCollection* to indicate if a card added/played should affect the score.
2	(a)	The sequence only allows cards to be added to the end and taken from the same end which is a LIFO structure **[1]**… and a stack is a LIFO structure that would be appropriate **[1]**… For the discard pile, sometimes you need to peek at the whole stack but generally just play cards to the top and then the whole pile is shuffled back in **[1]**… a stack could be suitable for this	4 marks	1 mark for each point (MAX. 4) **A:** stack for discard pile. **R:** queue for either.

Question		Suggested Solution	Total Marks	Marking Guidance
3	(a)	Splitting the deck into two halves (or using pointers to do the same thing) **[1]**… Choosing a number of cards from one half and add them to the combined deck (**A:** circular deck with counting solutions) **[1]**… Choosing a number of cards from the other half and add them to the combined deck **[1]**… Using a random number of cards (1 to 5, A. 1 to 5) **[1]**… Taking cards from the bottom of the split decks rather than the top. Repeating until the deck is fully combined **[1]**.	6 marks	1 mark for each point
	(b)	Most algorithms will have a space complexity twice that of the random shuffle that existed before, for storing the deck split and a new merged/combined deck.	1 mark	**A:** any version of the idea for 1 mark. **A:** circular array solutions with space complexity of the same as the storage for the deck as long as they are explained properly.
4	(a)		5 marks	1 mark for the Count column (**I:** spaces) 1 mark for a final return value of True 1 mark for the first value in the SequenceAsString column 1 mark for the last value in the SequenceAsString column 1 mark for the correct middle values in the SequenceAsString column (between the first and last value) **DPT:** -1 only for a missing space

Count	SequenceAsString	Return value
	""	
5	"P a"	
4	", P a"	
	"F a, P a"	
3	", F a, P a"	
	"P a, F a, P a"	True

	(b)		3 marks	1 mark for each column **DPT:** -1 only for a missing space (note that this is across parts (a) and (b) combined, total of -1 for a missing space across the two parts)

Count	SequenceAsString	Return value
	""	
5	"P a"	
4	", P a"	

Question		Suggested Solution	Total Marks	Marking Guidance
5	(b)	**Either:** Once a lock is solved, remove it from the list of locks so that it cannot be selected again when a new random lock is chosen. **Or:** Once a lock is solved, iterate through all the challenges and set the attribute `Met` for each to `False`.	2 marks	2 marks available for either solution as long as the details are clearly explained, otherwise award 1 mark **A:** any reasonable solution including the idea of moving `LockSolved` into the `Lock` class and checking it when choosing a new lock.
6	(a)	As the number of cards in the deck gets smaller then there will be more swaps than possible arrangements of the cards, which makes the additional swaps redundant and inefficient **[1]**…For example, with 2 cards there are only two combinations and with 6 only 720 but there are still 10000 swaps **[1]** …There is also the chance of the algorithm causing an exception with 0 or 1 cards but there is no check for this **[1]**.	3 marks	Award 1 mark for each point **A:** any expression of the idea that 10000 is more swaps than you need, which makes the extra ones unnecessary for the first mark. **A:** any reference or example to decreasing combinations for the second mark.
	(b)	The number of swaps could be a measure of deck size or could be set to a lower threshold when the deck is small **[1]**. This would be done by using a selection statement and setting a swaps variable to a lower value if the deck is short or to 10000 as now if the deck is large **[1]**.	2 marks	Award 1 mark for each point **A:** any expression of each concept for each mark. **A:** any other reasonable suggestions that would give 10000 for large decks and a much lower number for small decks.
7	(a)	When two arguments are supplied they are used to set the `ToolType` and `Kit` respectively **[1]**… The `CardNumber` is set using the next available `CardNumber` from the class/static variable `NextCardNumber` by the parent constructor when it is called **[1]**… In the case of a third argument being supplied, the parent constructor is not called and the `CardNumber` is set by the value of the parameter **[1]**.	3 marks	1 mark for each point (MAX. 3)

Question		Suggested Solution	Total Marks	Marking Guidance
8	(b)	A class or static variable has the same value for every object and is changed in them all when it changes in one **[1]**… An attribute may start off the same but has a different value in each object and if changed in one will not affect the others **[1]**.	2 marks	1 mark for each point
9	(a)	Inheritance is where a child gains the attributes and behaviour methods of its parent.	1 mark	**A:** other words with similar meanings.
	(b)	Aggregation associate is where one class contains another class but their lifespans are not linked and they can function independently.	1 mark	**R:** answers that do not refer to lifespan in some way.
	(c)	A data structure for which the memory usage will shrink and grow over time according to the storage needs.	1 mark	
10	(a)	***Solution #1*** Code **[1]**: **self.__MoveCard(self.__Deck, self.__Discard, self.__Deck.GetCardNumberAt(0))** … because *CardCollection* treats *DifficultyCards* and *ToolCards* **[1]** as cards **[1]**… which is an example of polymorphism but when the statement resolves any methods will actually execute in the child's object class **[1]**. ***Solution #2*** Code **[1]**: **def Process(self, Deck, Discard, Hand, Sequence, CurrentLock, Choice, CardChoice):** ** ChoiceAsInteger = None** … because this overrides the *Process* method **[1]**… in the parent *Card* class **[1]**… which will mean that *DifficultyCards* can be treated as cards but behave as themselves **[1]**. *Solution #3*	4 marks	For any one possible answer: 1 mark for each point and 1 mark for the code **A:** any example of code related to inheritance for 1 mark provided the explanation gains at least 1 mark. **R:** code only with no explanation.

Question		Suggested Solution	Total Marks	Marking Guidance
11		Create a new AdvancedLock class **[1]**… that inherits from Lock **[1]**… and override the GetLockSolved method **[1]**… so that when the basic challenges are solved it unlocks the final challenge and returns False instead of True **[1]**… this will then mean that it can refer to a [illegible] attribute such as SecretUnlocked **[1]**… when c[illegible]g GetLockSolved the next ti[illegible]	6 marks	1 mark for each point
12		The 'K [illegible]' card is in position 2 meaning that when GetCardFromDeck is called, CardChoice will be [illegible] **[1]**… The card in position 2 will have already been m[illegible]d from the hand to the deck as Mov[illegible]rd is called by PlayCardToSequence before GetC[illegible]FromDeck **[1]**… The hand is [illegible]played the player enters 3 to choose 'K c' as the ke[illegible] discard because it is now in po[illegible]n 3 **[1]**… When Process is called CardChoice is 2 [illegible] Choice is 3 **[1]**… Choice is [illegible]verted from a string to an integer, range checked ([illegible]) successfully and stored in ChoiceAsInteger **[1]**… as 3 >= 2, ChoiceAsInteger is decrem[illegible]nted by 1 **[1]**… and as ChoiceAsIn[illegible]er is still > 0 it is decremented by 1 again to give 1 [illegible] The selection statement ch[illegible]ck[illegible] against the card in the hand at index ChoiceAsInt[illegible]ger (which is 1) and that card is now [illegible] b' so the condition is False **[1]**… and 5 cards are d[illegible]rded from the deck to the discard p[illegible] **[1]**.	8 marks	1 mark for e[illegible]ch point (MAX. 8)
13	(a)	Using a constan[illegible] **[1]**… which would be declared once at the top of progr[illegible]m and could be changed in that si[illegible] place **[1]**.	2 marks	1 mark for each poi[illegible]
	(b)	It is possible to mis[illegible]t values **[1]** ,… update the wrong values **[1]**,… or m[illegible]e values **[1]**… constants have [illegible] identifier, making a value easier to understand **[1]**.	2 marks	1 mark for each point ([illegible]. 2) **A:** opposite points.
14	(a)	File handling can always generate exceptions **[1]**… because files could be locked **[1]**… removed/unavailable/inaccessible **[1]**	2 marks	1 mark for each point (MAX. 2)
	(b)	Converting an inputted string to an integer **[1]**… because if it fails you want to catch the error and ask the user to input again **[1]**.	2 marks	1 mark for each point **A:** any example of validation.

BREAKTHROUGH

Programming Tasks (Mark Sch

Task 1

Coding

- Printing out the nu ards left in the deck correctly each turn [1 mark]

Example Sol

```
print(self.__Sequence.GetCardDisplay())
# CODE ADDED
print(f"There are {self.__Deck.GetNumberOfCards()} cards left i
# END ADDITION
print(self.__Hand.GetCardDisplay())
```

Testing:

- Printing out the number of cards left correctly between SEQUENCE and HAND

```
SEQUENCE: empty

There are 33 cards left in the deck.

HAND:
```

Task 2

Coding:

- Changing GetChoice to show Peek (even if it doesn't check GetPeekUsed) [1 m
- Changing PlayGame to accept 'P' and printing out the three cards in the deck (re output) [1 mark]
- Adding the PeekUsed attribute with get/set methods to Lock [1 mark]

Example Solution

Changes to GetChoice

```
    def __GetChoice(self):
        pr
        # CHANGE
        if self.__CurrentLock.GetPeekUsed():
          Choice = input("(D)iscard inspect, (U)se card:> ").up
        else:
          Choice = input("(D)iscard inspect, (U)se card, (P)eek
        # END CHANGE
        return Choice
```

Changes to PlayGame

```
          print(self.__Discard.GetCardDisplay())
        # CODE ADDED
        elif MenuChoice == "P":
          if not self.__CurrentLock.GetPeekUsed():
            cards = ""
            for i in range(3):
              cards += self.__Deck.__Cards.DescriptionAt(i)+", "
            print("The next cards are: "+ cards[:-2])
            self.__CurrentLock.SetPeekUsed()
        # END ADDITION
        elif MenuChoice == "U":
```

Changes to Lock

```
    def __init__(self):
        self._Challenges = []
        # CODE ADDED
        self.__PeekUsed = False
        # END ADDITION
```

...

```
    # CODE ADDED
    def GetPeekUsed(self):
      return self.__PeekUsed

    def SetPeekUsed(self):
      self.__PeekUsed = True
    # END ADDITION
```

Testing:

- Peek is an option, works correctly and then disappears [1 mark] ↓

```
(D)iscard inspect, (U)se card, (P)eek:> p
The next three cards are: F a, F c, K a

Current score: 1

CURRENT LOCK
------------
Not met:        P c, F c, K c


SEQUENCE:

------
| P c |
------


HAND:

-----------------------------
| P a | K b | P b | P a | K b |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 1
(D)iscard or (P)lay?:> d

Current score: 1
```

- Peek reappears for the next lock and works and then disappears and doesn't wo

```
(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 1
(D)iscard or (P)lay?:> p

A challenge on the lock

Lock has [illegible] Your score is now: 21

Current score: 21

CURRENT LOCK
------------
Not met:        P a, F a, P a
Not met:        P c, F c, P c


SEQUENCE:

-----------------
| P c | F c | K c |
-----------------


HAND:

-----------------------------
| K b | K b | K a | F a | P c |
-----------------------------


(D)iscard [illegible] (P)eek:> p
The next [illegible]: D1f, P b, F b

Current score: 21
```

```
(D)iscard inspect, (U)

Current score: 21

CURRENT LOCK
------------
Not met:        P a, F
Not met:        P c, F
```

Task 3

Coding:

- Checking for correct condition to print out the error [1 mark]
- Printing out a sensible error message with the card or tool type that is in error [1

Example Solution

Changes to `PlayCardToSequence`

```
            self.__GetCardPeek(CardChoice)
        # CODE
        e:
        int(f"ERROR: The card you are trying to play
        ({self.__Hand.GetCardDescriptionAt(CardChoice - 1)
        last card in the sequence.")
        # END ADDITION
    else:
```

Testing:

- Showing the error message and the hand and sequence afterwards confirming t discarded [1 mark] ↓

Task 4

Coding:

- Printing out the correct message only when a mulligan is available [1 mark]
- Adding the `MulliganUsed` attribute to `Breakthrough` and initialising it to False
- Implementing the mulligan to add all the cards from the player's hand, the discard [1 mark]
- Shuffling up and dealing again (and discarding any difficulty cards drawn) [1 ma

Example Solution

Changes to `GetChoice`

```
    def __GetChoice(self):
        print()
        # CHANGE
        if self.__MulliganUsed:
          Choice = input("(D)iscard inspect, (U)se card:> ").up
        else:
          Choice = input("(D)iscard inspect, (U)se card, (M)ull
        # END CHANGE
        return Choice
```

Changes to `Breakthrough`

```
        self.__LoadLocks()
        # CODE ADDED
        self.__MulliganUsed = False
        # END ADDITION
```

Changes to `PlayGame`

```
if MenuChoice == "D":
    print(self.__Di[illegible]GetCardDisplay())
# CODE ADD[illegible]
elif MenuC[illegible] == "M":
    if not self.__MulliganUsed:
      # move cards from sequence to deck
      for Count in range(self.__Sequence.GetNumberOfCards()):
        self.__MoveCard(self.__Sequence, self.__Deck, self.__Se
      # move cards from discard pile to deck
      for Count in range(self.__Discard.GetNumberOfCards()):
        self.__MoveCard(self.__Discard, self.__Deck, self.__Dis
      # move cards from hand to deck
      for Count in range(self.__Hand.GetNumberOfCards()):
        self.__MoveCard(self.__Hand, self.__Deck, self.__Hand.Ge
      # shuffle up and deal
      self.__Deck.Shuffle()
      for Count in range(5):
        while self.__Deck.GetCardDescript[illegible] == "Dif":
          self.__MoveCard(self.__De[illegible], self.__Discard, self.__De
        self.__MoveCard(self[illegible], self.__Hand, self.__Deck.Ge
      self.__MulliganU[illegible]
# END ADDITION
elif MenuC[illegible] :
```

Testing:

- Showing a mulligan being used after solving a challenge [1 mark] ↓

```
CURRENT LOCK
------------
Not met:       P a, F a, P a
Not met:       P b, F b, P b
Challenge met: K c


SEQUENCE:

------
| K c |
------

HAND:

-----------------------------
| K a | F b | P b | K a | P b |
-----------------------------

(D)iscard inspect, (U)se card, (M)ulligan:> m

Current score: 8

CURRENT LOCK
------------
Not met:       P a, F a, P a
Not met:       P b, F b, P b
Challenge met: K c


SEQUENCE: empty

HAND:

-----------------------------
| K b | F a | K a | P b | F c |
-----------------------------

(D)iscard inspect, (U)se card:>
```

- Showing an attempt to use the mulligan again failing [1 mark] →

Task 5

Coding:

- Printing out quit as a menu option and including it in the selection statement in P
- Cleanly exiting the main game loop in `PlayGame` without using `system.exit`, g and successfully ending the program [1 mark]

Example Solution

Changes to `GetChoice`

```
    def __GetChoice(self):
        print()
        # CHANGE
        Choice = input("(D)iscard inspect, (U)se card, (Q)uit:>
        # END CHANGE
        return Choice
```

Changes to `PlayGame`

```
            self.__SetupGame()
            # CODE ADDED
            hasQuit = False
            # END ADDITION
            while not self.__GameOver:
                self.__LockSolved = False
                while not self.__LockSolved and not self.__Game
                    print()
                    print("Current score:", self.__Score)
                    print(self.__CurrentLock.GetLockDetails())
                    print(self.__Sequence.GetCardDisplay())
                    print(self.__Hand.GetCardDisplay())
                    MenuChoice = self.__GetChoice()
                    if MenuChoice == "D":
                        print(self.__Discard.GetCardDisplay())
                    # CODE ADDED
                    elif MenuChoice == "Q":
                      hasQuit = True
                    # END ADDITION
                    elif MenuChoice == "U":
                        CardChoice  = self.__GetCardChoice()
                        DiscardOrPlay = self.__GetDiscardOrPlay
                        if DiscardOrPlay == "D":
                            self.__MoveCard(self.__Hand, self.__
                            self.__Hand.GetCardNumberAt(CardCho
                            self.__GetCardFromDeck(CardChoice)
                        elif DiscardOrPlay == "P":
                            self.__PlayCardToSequence(CardChoic
                    if self.__CurrentLock.GetLockSolved():
                        self.__LockSolved = True
                        self.__ProcessLockSolved()
                # ADDED the if and moved the existing code to t
                if hasQuit:
                  self.__GameOver = True
                  self.__Score += self.__Deck.GetNumberOfCards(
                  print("Final score:", self.__Score)
                else:
                  self.__GameOver = self.__CheckIfPlayerHasLost
                # END CHANGE
        else:
```

Testing:

- Printing out a final score of 33 (if a Pick was played), 34 (if a File was played) or 3

```
HAND:

-----------------------------
| P b | P a | F c | K b | K c |
-----------------------------


(D)iscard inspect, (U)se card, (Q)uit:> u
Enter a number beg[illegible] to specify card to use:> 2
(D)iscard [illegible]

Current [illegible]: 1

CURRENT LOCK
------------
Not met:      P a, F a, P a
Not met:      K b


SEQUENCE:

------
| P a |
------


HAND:

---------------------------
| P b | F c | K b | [illegible]
---------------------------


(D)iscard inspect, (U)se card, (Q)uit:> q
Final score: 33
```

Task 6

Coding:

- Adding the three attributes `numPicks`, `numKeys` and `numFiles` to the `CardColl` to 0 [1 mark]
- Ensuring that at least one attribute is updated correctly when a card is added [1
- Ensuring that all three attributes are updated correctly when a card is removed [
- Creating a `DisplayStats` method that will print out the percentage of each type (even if not to two decimal places). Note that correctly means dividing the num of cards in the deck. [1 mark]

Example Solution

Changes to the CardCollection class

```
        # CODE ADDED
        self.__numPicks = 0
        self.__numFiles = 0
        self.__numKeys = 0
        # END ADDITION
```

```
    def AddCard(self, C):
      # CODE ADDED
      if C.GetDescription()[0] == "F":
        self.__numFiles += 1
      elif C.GetDescription()[0] == "K":
        self.__numKeys += 1
      elif C.GetDescription()[0] == "P":
        self.__numPicks += 1
      # END ADDITION
      self._Cards.append(C)

    def RemoveCard(self, CardNumber):
      CardFound = False
      Pos = 0
      while Pos < len(self._Cards) and not CardFound:
        if self._Cards[Pos].GetCardNumber() == CardNumber:
          CardToGet = self._Cards[Pos]
          CardFound = True
          self._Cards.pop(Pos)
        Pos += 1
      # CODE ADDED
      if CardToGet.GetDescription()[0] == "F":
        self.__numFiles -= 1
      elif CardToGet.GetDescription()[0] == "K":
        self.__numKeys -= 1
      elif CardToGet.GetDescription()[0] == "P":
        self.__numPicks -= 1
      # END ADDITION
      return CardToGet

    # CODE ADDED
    def DisplayStats(self):
      keyChance = self.__numKeys/self.GetNumberOfCards() * 100
      pickChance = self.__numPicks/self.GetNumberOfCards() * 100
      fileChance = self.__numFiles/self.GetNumberOfCards() * 100
      print("There is a {keyChance:0.2f}% chance that the next
      {fileChance:0.2f}% chance that it will be a file and a {p
      that it will be a pick.")
    # END ADDITION
```

Changes to GetCardFromDeck

```
                print(self.__Hand.GetCardDisplay())
                # CODE ADDED
                self.__Deck.DisplayStats()
                # END ADDITION
                print("To deal with this you need to either loss
```

Testing:

- Showing the percentage of at least one tool (even if incorrect) to two decimal pla
  which card the player would like to select or whether to discard from the deck.

  Note that the percentages are unlikely to match the ones below. [1 mark] ↓

```
Difficulty encountered!

HAND:

------------------------
| P b | F b | P c | P b |
------------------------

There is a 28.57% chance that the next card will be a key, a 21.43% ch
 it will be a file and a 35.71% chance that it will be a pick.
To deal with this you need to either lose a key (enter 1-5 to specify
of key) or (D)iscard five cards from the deck:>
```

Task 7

Coding:

- Adding one multi-tool of each kind to the deck at creation time [1 mark]
- Adding one multi-tool of each kind to the deck whenever a lock is solved [1 mark]
- Adding the two `SetCardToolkit` methods that successfully allow a card's toolkit
- Changing `PlayCardToSequence` to ask which toolkit the player would like when
- Calling `SetCardToolkit` for the correct card and toolkit from `PlayCardToSequence`

Example Solution

Changes to CreateStandardDeck

```
    def __CreateStandardDeck(self):
        # CHANGE
        NewCard = ToolCard("P", "m")
        self.__Deck.AddCard(NewCard)
        NewCard = ToolCard("F", "m")
        self.__Deck.AddCard(NewCard)
        NewCard = ToolCard("K", "m")
        self.__Deck.AddCard(NewCard)
        # END CHANGE
        for Count in range(5):
```

Changes to `ProcessLockSolved`

```
    self.__MoveCard(self.__Discard, self.__Deck, self.__Discard
        # CHANGE
        NewCard = ToolCard("P", "m")
        self.__Deck.AddCard(NewCard)
        NewCard = ToolCard("F", "m")
        self.__Deck.AddCard(NewCard)
        NewCard = ToolCard("K", "m")
        self.__Deck.AddCard(NewCard)
        # END CHANGE
        self.__Deck.Shuffle()
```

Changes to `PlayCardToSequence`

```
    def __PlayCardToSequence(self, CardChoice):
      # CHANGE
      if self.__Hand.GetCardDescriptionAt(CardChoice -1)[2] ==
        toolkit = input("Which toolkit would you like to choose
        self.__Hand.SetCardToolkit(CardChoice -1, toolkit)
      # END CHANGE
      if self.__Sequence.GetNumberOfCards() > 0:
```

Creation of `SetCardToolkit` in `CardCollection`

```
    def SetCardToolkit(self, pos, kit):
      if self._Cards[pos].GetDescription()[2] == "m":
        self._Cards[pos].SetCardToolkit(kit)
```

Creation of SetCardToolkit in ToolCard

```
    def SetCardToolkit(self, kit):
        self._Kit = kit
```

Testing:

- Showing the sequence updated with the card played of the `toolkit` chosen [1 s

```
SEQUENCE:

-----------------------------
| P b | K b | F b | P a | F a |
-----------------------------


HAND:

-----------------------------
| F c | P c | P b | P b | K c |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 3
(D)iscard or (P)lay?:> p
Which toolkit would you like to choose? a

A challenge on the lock has been met.


Current score: 30

CURRENT LOCK
------------
Challenge met: P a, F a, P a
Not met:       K b


SEQUENCE:

-----------------------------------
| P b | K b | F b | P a | F a | P a |
-----------------------------------


HAND:

-----------------------------
| F c | P c | P b | K c | P a |
-----------------------------
```

Task 8

Coding:

- Changing PlayGame to pass in the argument for the sequence to GetLockDeta
  GetLockDetails to accept the new parameter [1 mark]
- Changing GetLockDetails to match a single card on the sequence to the first
  message to partially met, also to not crash when there is an empty sequence [
- Changing GetLockDetails to generate a partially met message when only the
  the first card of a challenge [1 mark]
- Changing GetLockDetails to generate a partially met message when the last
  the first two cards of a challenge. [1 mark]

Example Solution

Changes to GetLockDetails

```
    def GetLockDetails(self, sequence):
    ...
    ...

            else:
                # CHANGE
                condition = C.GetCondition()
                if len(condition) == 3:
                  seqLen = sequence.GetNumberOfCards() - 1
                  if seqLen > 0 and condition[1] == sequence.Get
                  and condition[0] == sequence.GetCardDescriptio
                    LockDetails += "Partially met: "
                  elif seqLen >= 0 and condition[0] == sequence.G
                    LockDetails += "Partially met: "
                  else:
                    LockDetails += "Not met:       "

                    LockDetails += "Not met:       "
                # END CHANGE
            LockDetails += self.__ConvertConditionToString(C.Get
```

Changes to PlayGame

```
                    print("Current score:", self.__Score)
                    # CHANGE
                    print(self.__CurrentLock.GetLockDetails(sel
                    # END CHANGE
                    print(self.__Sequence.GetCardDisplay())
```

Testing:

- First card played to sequence doesn't generate partially met (if it doesn't match)

```
CURRENT LOCK
------------
Not met:        P a, F a, P a
Not met:        P c, F c, P c


SEQUENCE:

------
| F b |
------

HAND:

----------------------------
| P a | P c | F b | P b | F b |
----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 1
(D)iscard or (P)lay?:> p

Current score: 3

CURRENT LOCK
------------
Partially met: P a, F a, P a
Not met:        P c, F c, P c


SEQUENCE:

------------
| F b | P a |
------------

HAND:

----------------------------
| P c | F b | P b | F b | P c |
----------------------------
```

- Last two cards on the sequence matching first two of a challenge generates partially met [1 mark] →

```
HAND:

----------------
| P c | K b | P c
----------------


(D)iscard inspect
Enter a number be
(D)iscard or (P)l

Current score: 3

CURRENT LOCK
------------
Partially met: P
Not met:        P


SEQUENCE:

------------
| F b | P a |
------------

HAND:

----------------
| P c | K b | P c
----------------


(D)iscard inspect
```

Task 9

Coding:

- Adding a variable for bonusCounter and initialising it to 0 for each new lock [1 m
- Adding 1 to the variable each time a card is played or discarded [1 mark]
- Awarding the correct bonus once the lock is solved (including 0 if over 20 cards v variable to 0 [1 mark]

Example Solution

```
def PlayGame(self):
    if len(self.__Locks) > 0:
        self.__SetupGame()
        while not self.__GameOver:
            self.__LockSolved = False
            # CODE ADDED
            bonusCounter = 0
            # END ADDITION
            while not self.__LockSolved and not self.__GameOver
                print()
                print("Current score:", self.__Score)
                print(self.__CurrentLock.GetLockDetails())
                print(self.__Sequence.GetCardDisplay())
                print(self.__Hand.GetCardDisplay())
                MenuChoice = self.__GetChoice()
                if MenuChoice == "D":
                    print(self.__Discard.GetCardDisplay())
                elif MenuChoice == "U":
                    CardChoice = self.__GetCardChoice()
                    DiscardOrPlay = self.__GetDiscardOrPlayChoic
                    # CODE ADDED
                    bonusCounter += 1
                    # END ADDITION
                    if DiscardOrPlay == "D":
                        self.__MoveCard(self.__Hand, self.__Disc
                        self.__Hand.GetCardNumberAt(CardChoice
                        self.__GetCardFromDeck(CardChoice)
                    elif DiscardOrPlay == "P":
                        self.__PlayCardToSequence(CardChoice)
                if self.__CurrentLock.GetLockSolved():
                    self.__LockSolved = True
                    self.__ProcessLockSolved()
                    # CODE ADDED
                    bonus = max(0,20-bonusCounter)
                    self.__Score += bonus
                    print("This locked awarded you",bonus,"bonu
                    bonusCounter = 0
                    # END ADDITION
            self.__GameOver = self.__CheckPlayerHasLost()
    else:
        print("No locks in play")
```

Testing:

- Solving a lock in under 20 cards and getting the correct bonus (which doesn't ha
  [1 mark] ↓

```
A challenge on the lock has been met.

Lock has been solved.  Your score is now: 21
This lock awarded you 17 bonus points.

Current score: 38

CURRENT LOCK
------------
Not met:        F c, P c, F c


SEQUENCE:

-----------------
| P b | K b | F b |
-----------------


HAND:

-----------------------------
| P a | P a | P b | K c | F c |
-----------------------------


(D)iscard inspect, (U)se card:
```

- Solving a lock over 20 cards and getting 0 bonus. [1 mark] ↓

```
A challenge on the lock has been met.

Lock has been solved.  Your score is now: 38
This lock awarded you 0 bonus points.

Current score: 38

CURRENT LOCK
------------
Not met:        P a, F a, K a
```

Task 10

Coding:

- Modifying SetupGame to have a 25% chance of adding a GeniusCard [1 mark]
- Modifying ProcessLockSolved to have a 25% chance of adding a GeniusCard
- Creating a GeniusCard class that inherits from Card, has a constructor with set the CardType to Gen [1 mark]
- Asking the user to enter a challenge number or discard when a genius card is dr
- Processing the GeniusCard correctly if the challenge chosen [1 mark]
- Processing the GeniusCard correctly to discard it [1 mark]
- Handling the discarding of GeniusCard correctly if drawn while refilling the ha and print a message [1 mark]

Example Solution

Creation of AddGeniusCardToDeck

```
    def __AddGeniusCardToDeck(self):
        self.__Deck.AddCard(GeniusCard())
```

Creation of GeniusCard

```
# CODE ADDED
class GeniusCard(Card):
    def __init__(self):
        self._CardType = "Gen"
        super().__init__()

    def GetDescription(self):
        return self._CardType

    def Process(self, CurrentLock, Choice):
        CurrentLock.SetChallengeMet(Choice-1,True)

# END ADDITION
```

Changes to SetupGame

```
            self.__AddDifficultyCardsToDeck()
            # CODE ADDED
            if random.randint(1,4) == 1:
                self.__AddGeniusCardToDeck()
            # END ADDITION
            self.__Deck.Shuffle()
```

Changes to ProcessLockSolved

```
            self.__MoveCard(self.__Discard, self.__Deck, self.__
        # CODE ADDED
        if random.randint(1,4) == 1:
          self.__AddGeniusCardToDeck()
        # END ADDITION
        self.__Deck.Shuffle()
```

Changes to GetCardFromDeck

```
                CurrentCard.Process(self.__Deck, self.__Discard
                self.__Sequence, self.__CurrentLock, Choice, Car
            # CODE ADDED
            elif self.__Deck.GetCardDescriptionAt(0) == "Gen":
                CurrentCard = self.__Deck.RemoveCard(self.__Dec
                print()
                print("Genius card encountered!")
                print("You can either use this card immediately to s
                it.")
                Choice = input(f"enter 1-{self.__CurrentLock.Get
                solve a challenge or (D)iscard it so it can come
                ").upper()
                if Choice.strip() == "D":
                    self.__Discard.AddCard(CurrentCard)
                else:
                    CurrentCard.Process(self.__CurrentLock, int(
            # END ADDITION
        while self.__Hand.GetNumberOfCards() < 5 and self.__Dec

            if self.__Deck.GetCardDescriptionAt(0) == "Dif":
                self.__MoveCard(self.__Deck, self.__Discard, sel
                print("A difficulty card was discarded from the
                hand.")
            # CODE ADDED
            elif self.__Deck.GetCardDescriptionAt(0) == "Gen":
                self.__MoveCard(self.__Deck, self.__Discard, sel
                print("A genius card was discarded from the deck
            # END ADDITION
            else:
```

Testing:

- Using a GeniusCard [illegible] specifically [1 mark]

```
CURRENT LOCK
------------
Not met:      P a, F a, P a
Not met:      P b, F b, P b
Not met:      K c


SEQUENCE:

-----------------
| P c | F c | K c |
-----------------


HAND:

-----------------------------
| K a | K a | P b | P a | F a |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number [illegible]een 1 and 5 to specify
(D)iscar[illegible] ?:> p

[illegible] encountered!
You can either use this card immediately [illegible]
enter 1-3 to solve a challenge or (D)iscar[illegible]

Current score: 22

CURRENT LOCK
------------
Not met:      P a, F a, P a
Challenge met: P b, F b, P b
Not met:      K c
```

- Discarding a GeniusCard successfully [1 mark] ↓

```
Genius card encountered!
You can either use this card immediately to solve a challenge or discard it,
enter 1-3 to solve a challenge or (D)iscard it so it can come up after reshu

Current score: 30

CURRENT LOCK
------------
Challenge met: P a, F a, P a
Challenge met: P b, F b, P b
Not met:       K c


SEQUENCE:

-----------------------------------
| P c | F c | K c | P a | F a | P a |
-----------------------------------


HAND:

-----------------------------
| K a | K b | P b | K c | P b |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use
```

Task 11

Coding:

- Adding the `Credits` attribute and initialising it to 10 [1 mark]
- Asking whether the player would like to buy a tool only when they have played c least 2 credits left [1 mark]
- Ensuring that keys are not listed if they have few[illegible] credits remaining (eve mark) [1 mark]
- Adding a tool card of the correct [illegible] toolkit to the player's hand in the fifth p
- Removing the tool card [illegible] ck at the correct position [1 mark]
- Deducting [illegible] ber of credits for buying a card [1 mark]
- Printing ou[illegible] of the correct number of each tool available and not printing to [1 mark]
- Printing option 10 correctly at the end of the menu, once and once only [1 mark]
- Having an iteration statement to correctly calculate the number of tools of each t
- Returning a list from `PrintToolsAvailable` that contains the index of a card w [1 mark]

Example Solution

Adding the `Credits` attribute

```
        self.__LoadLocks()
        # CODE ADDED
        self.__Credits = 10
        # END ADDITION
```

Changes to `GetCardFromDeck`

```
        if self.__Deck.GetNumberOfCards() > 0:
            # CODE
            [illegible]e[illegible].__Credits >= 2:
                Choice = input("Would you like to buy a tool (y
                if Choice.strip().lower() == "y":
                    ToolList = self.__Deck.PrintToolsAvailable(
                    else False)
                    CardChosen = int(input("Which tool would yo
                    if CardChosen != 10 and ToolList[CardChosen
                        self.__MoveCard(self.__Deck, self.__Han
                        self.__Deck.GetCardNumberAt(ToolList[Ca
                        if CardChosen > 6:
                            self.__Credits -= 3
                        else:
                            self.__Credits -= 2
            # END ADDITION
            if self.__Deck.GetCardDescriptionAt(0) == "Dif":
```

New PrintToolsAvailable method in CardCollection

```
    def PrintToolsAvailable(self, KeysAvailable):
        Tools = ["F a", "F b", "F c", "P a", "P b", "P c", "K a
        ToolList = [-1,-1,-1,-1,-1,-1,-1,-1,-1]
        ToolsAvailable = [0,0,0,0,0,0,0,0,0]
        for i in range(self.GetNumberOfCards()):
            if self._Cards[i].GetDescription() == "F a":
                ToolsAvailable[0] += 1
                if ToolList[0] ==
                    ToolList[0]
            elif self._Cards[i].GetDescription() == "F b":
                ToolsAvailable[1] += 1
                if ToolList[1] == -1:
                    ToolList[1] = i
            elif self._Cards[i].GetDescription() == "F c":
                ToolsAvailable[2] += 1
                if ToolList[2] == -1:
                    ToolList[2] = i
            elif self._Cards[i].GetDescription() == "P a":
                ToolsAvailable[3] += 1
                if ToolList[3] == -1:
                    ToolList[3] = i
            elif self._Cards[i].GetDescription() == "P b":
                ToolsAvailable[4] += 1
                if ToolList[4] == -1:
                    ToolList[4] = i
            elif self._Cards[i].GetDescription() == "P c":
                ToolsAvailable[5] += 1
                if ToolList[5] == -1:
                    ToolList[5] = i
            elif self._Cards[i].GetDescription() == "K a" and Ke
                ToolsAvailable[6] += 1
                if ToolList[6] == -1:
                    ToolList[6] = i
            elif self._Cards[i].GetDescription() == "K b" and Ke
                ToolsAvailable[7] += 1
                if ToolList[7] == -1:
                    ToolList[7] = i
            elif self._Cards[i].GetDescription() == "K c" and Ke
                ToolsAvailable[8] += 1
                if ToolList[8] == -1:
                    ToolList[8] = i

        for i in range(len(Tools)):
            if ToolList[i] >= 0:
                print(f"{i+1}. {Tools[i]} ({ToolsAvailable[i]}

        print("10. No Tool (buy nothing)")

        return ToolList
```

Testing:

- Buying two tools [1 mark] ↓

```
Enter L to load a game from a file, anything else to play a new game:>

Current score: 0

CURRENT LOCK
------------
Not met:       P a, F a, P a
Not met:       K b



SEQUENCE: empty


HAND:

-----------------------------
| P c | K a | P c | P c | K c |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 3
(D)iscard or (P)lay?:> d
Would you like to buy a tool (y/n)? y
1. F a (3 available)
2. F b (3 available)
3. F c (3 available)
4. P a (5 available)
5. P b (5 available)
6. P c (2 available)
7. K a (2 available)
8. K b (3 available)
9. K c (2 available)
10. No Tool (buy nothing)
Which tool would you like to buy? 4

Current score: 0

CURRENT LOCK
------------
Not met:       F b, P a
Not met:
```

```
SEQUENCE: empty



HAND:

-----------------
| P c | K a | P c
-----------------


(D)iscard inspect
Enter a number be
(D)iscard or (P)l
Would you like to
1. F a (3 availab
2. F b (3 availab
3. F c (3 availab
4. P a (4 availab
5. P b (5 availab
6. P c (2 availab
7. K a (2 availab
8. K b (3 availab
9. K c (2 availab
10. No Tool (buy
Which tool would

Current score: 1

CURRENT LOCK
------------
Not met:       P
Not met:       K



SEQUENCE:

------
| P a |
------


HAND:

-----------------
| P c | K a | P c
```

- Trying to buy a tool with 2 credits left [1 mark] ↓

```
Would you like to buy a tool (y/n)? y
1. F a (2 available)
2. F b (3 available)
3. F c (1 available)
4. P a (4 available)
5. P b (4 available)
6. P c (3 available)
10. No Tool (buy nothing)
```

Task 12

Coding:

- Selection statement to check whether the challenge just solved was at least thre
- Iteration statement to run once for each tool card in the challenge just solved [1 m
- Call to `MoveCard` to move one tool card from the sequence to the discard pile in [1 mark]

Example Solution

```
if self.__CurrentLock [illegible] ConditionMet(SequenceAsString):
    # CODE [illegible]
    if len[illegible]nceAsString) >= 13:
      for i[illegible] range(len(SequenceAsString) // 4):
        self.__MoveCard(self.__Sequence, self.__Discard, self.__S
    # END ADDITION
    return True
```

Testing:

- Solve a challenge with one card, then with three cards [1 mark] ↓

```
CURRENT LOCK
------------
Not met:       P a, F a, P a
Challenge met: K b


SEQUENCE:

-----------------
| K b | P a | F a |
-----------------


HAND:

-----------------------------
| F b | K a | K c | K b | P a |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 5
(D)iscard or (P)lay?:> p

A challenge on the lock has been met.

Lock has been solved.  Your score is now: 27

Current score: 27

CURRENT LOCK
------------
Not met:       F c, P c, F c


SEQUENCE:

------
| K b |
------
```

Task 13

Coding:

- Changing `GetChoice` to correctly prompt you to (S)ave the game and `PlayGame` if 'S' was chosen [1 mark]
- Returning a string of the correct format for the save file from `GetChallengesAs`
- Returning a string of the correct format for the sa[illegible]e [illegible]m `GetChallengesMe`
- Saving the current score to the save ga[illegible] [1 mark]
- Saving the current lock to the [illegible] [1 mark]
- Saving the hand, seq[illegible] [illegible]k and discard pile to the save game file [1 mark]
- Having a [illegible] loop that creates the string for a `CardCollection` in the [illegible] file [1 ma[illegible]

Example Solution

Changes to `GetChoice`

```
    def __GetChoice(self):
        print()
        # CHANGE
        Choice = input("(D)iscard inspect, (S)ave or (U)se card
        # END CHANGE
        return Choice
```

Changes to `PlayGame`

```
                    print(self.__Discard.GetCardDisplay())
                # CODE ADDED
                elif MenuChoice == "S":
                    self.__SaveGame()
                # END ADDED
                elif MenuChoice == "U":
```

Code for Sav[illegible]

```
    def __SaveGame(self):
      saveName = "game1.txt"

      try:
        with open(saveName,"w") as saveFH:
          print(self.__Score,file=saveFH)
          print(self.__CurrentLock.GetChallengesAsString(),file
          print(self.__CurrentLock.GetChallengesMetAsString(),f
          if self.__Hand.GetNumberOfCards() > 0:
            hand = ",".join([f"{self.__Hand.GetCardDescriptionA
            {self.__Hand.GetCardNumberAt(i)}" for i in
            range(self.__Hand.GetNumberOfCards())])
          else:
            hand = ""
          print(hand,file=saveFH)
          if self.__Sequence.GetNumberOfCards() > 0:
            seq = ",".j[illegible] f"{self.__Sequence.GetCardDescriptio
            {self.__Sequence.GetCardNumberAt(i)}" for i in
            range(self.__Sequence.GetNumberOfCards())])
          [illegible]
            seq = ""
          print(seq,file=saveFH)
          if self.__Discard.GetNumberOfCards() > 0:
```

```
            discard = ",".join([f"{self.__Discard.GetCardDescri
            {self.__Discard.GetCardNumberAt(i)}" for i in
            range(self.__Discard.GetNumberOfCards())])
          else:
            discard = ""
          print(discard,file=saveFH)
          if self.__Deck.GetNumberOfCards() > 0:
            deck = ",".join([f"{self.__Deck.GetCardDescriptionA
            {self.__Deck.GetCardNumberAt(i)}" for i in
            range(self.__Deck.GetNumberOfCards())])
          else:
            deck = ""
          print(deck,file=saveFH)
        print("File saved.")
      except:
        print("File not saved")
```

Code for GetChallengesAsString

```
    def GetChallengesAsString(self):
      challenges = ""
      for C in self._Challenges:
        if len(challenges) > 0:
          challenges += ";"
        challenges += self.__ConvertConditionToString(C)
      return challenges
```

Code for GetChallengesMetAsString

```
    def GetChallengesMetAsString(self):
      challenges = ""
      for C in self._Challenges:
        if len(challenges) > 0:
          challenges += ";"
        if self.__IsMet(C):
          challenges += "Y"
        else:
          challenges += "N"
      return challenges
```

Testing:

- Saving game then loading game [1 mark] ↓

```
(D)iscard inspect, (S)ave or (U)se card?:> s
File saved.

Current score: 8

CURRENT LOCK
------------
Not met:       K a
Challenge met: K b
Not met:

SEQUENCE:

------
| K b |
------

HAND:

-----------------------------
| P b | P c | P b | P c | F b |
-----------------------------

(D)iscard inspect, (S)ave or (U)se card?:>
```

```
Enter L to load a game from a file, anything else to play a new game:> l

Current score: 8

CURRENT LOCK
------------
Not met:       K a
Challenge met: K b
Not met:       K c

SEQUENCE:

------
| K b |
------

HAND:

-----------------------------
| P b | P c | P b | P c |
-----------------------------

(D)iscard inspect, (S)ave or (U)se card?:>
```

Task 14

Coding:

- Adding 5 to BonusPool after adding 5 to the score when completing a challenge
- Adding 5 to BonusPool when playing a card to the sequence that is a partial so
- Resetting BonusPool to 0 under all circumstances wher ard is not played to
- Creating the new attribute BonusPool and initial m, it 0 [1 mark]
- Writing the code for IsPartial such that it turns True if the card just played a didn't add to an existing challenge [1 mark]

Example Sol

Changes to P dToSequence

```
            print()
            # CHANGE
            self.__Score += 5 + self.__BonusPool
            # END CHANGE
            # CODE ADDED
            self.__BonusPool += 5
        else:
            if self.__CurrentLock.IsPartial(self.__Sequence):
              self.__BonusPool += 5
            else:
              self.__BonusPool = 0
        # END ADDITION
```

Changes to PlayGame

```
                    self.__C rumDeck(CardChoice)
                    # ADDED
                    self.__BonusPool = 0
                    # END ADDITION
                elif DiscardOrPlay == "P":
```

Code for IsPartial

```
    def IsPartial(self, seq):
      partial = False
      for C in self._Challenges:
        condition = C.GetCondition()
        if len(condition) == 3:
          seqLen = seq.GetNumberOfCards() - 1
          if seqLen > 0 and condition[1] == seq.GetCardDescripti
          condition[0] == seq.GetCardDescriptionAt(seqLen-1):
            partial = True
          elif seqLen >= 0 and condition[0] == seq.GetCardDescri
            partial = True
      return partial
```

Code for new BonusPool attribute

```
        self. Lond ()
        # A
        se onusPool = 0
        # END ADDITION
```

Testing:

- Playing three cards to solve a challenge then solve it one card after another [1 m

```
Enter L to load a game from a file, anything else to play a new game:>

Current score: 0

CURRENT LOCK
------------
Not met:      P a, F a, P a
Not met:      P b, F b, P b
Not met:      K c


SEQUENCE: empty


HAND:

-----------------------------
| P a | P a | K b | K b | F a |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 1
(D)iscard or (P)lay?:> p

Current score: 1

CURRENT LOCK
------------
Not met:      P a, F a, P a
Not met:      P b, F b, P b
Not met:      K c
```

```
SEQUENCE:

------
| P a |
------


HAND:

-----------------------------
| P a | K b |       | F c |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 4
(D)iscard or (P)lay?:> p

Difficulty encountered!

HAND:

-----------------------
| P a | K b | K b | F c |
-----------------------

To deal with this you need to either lose a key (enter 1-5 to specify position of key) or (D)iscard five c

Current score: 3

CURRENT LOCK
------------
Not met:      P a, F a, P a
Not met:      P b, F b, P b
Not met:      K c


SEQUENCE:

-----------
| P a | F a |
-----------
```

```
HAND:

-------------------------------
| P a | K b | F c | K c | K a |
-------------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 1
(D)iscard or (P)lay?:> p

A challenge on the lock has been met.


Current score:

CURRENT LOCK
------------
Challenge met: P a, F a, P a
Not met:       P b, F b, P b
Not met:       K c


SEQUENCE:

-------------------
| P a | F a | P a |
-------------------


HAND:

-------------------------------
| K b | F c | K c | K a | P c |
-------------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 3
(D)iscard or (P)lay?:> p

A challenge on the lock has been met.


Current score: 43
```

```
CURRENT LOCK
------------
Challenge met: P a, F
Not met:       P b, F
Challenge met: K c


SEQUENCE:

---------------------
| P a | F a | P a | K
---------------------


HAND:

---------------------
| K b | F c | K a | P
---------------------


(D)iscard inspect, (U
Enter a number betwee
(D)iscard or (P)lay?:

Current score: 44

CURRENT LOCK
------------
Challenge met: P a, F
Not met:       P b, F
Challenge met: K c


SEQUENCE:

---------------------
| P a | F a | P a | K
---------------------


HAND:

---------------------
| K b | K a | P c | F
---------------------
```

Task 15

Coding:

- Adding `FinalLock` as a private attribute and initialising it to 0 [1 mark]
- Adding the selection conditions for `FinalLock == 1` and `FinalLock == 2` to c they don't have the correct contents) [1 mark]
- Changing the condition of the selection statement in ge ar FromDeck correctly
- Changing `ProcessLockSolved` to have 1 ate s to find a soluble lock [1 ma
- Changing `ProcessLockSolved` erateSolubleLock once 10 attemp
- Changing `ProcessLo` set `FinalLock` to 1 and skip the main body c
- Writing Ge eLock such that it always generates a soluble lock (ru tools of th type excluded) [1 mark]
- Writing `GenerateChallenge` such that it generates a possible challenge from th not have two tools of the same type consecutively) [1 mark]
- Returning True and False correctly from `IsSoluble` [1 mark]

Example Solution

Addition of `FinalLock` attribute

```
        # CODE ADDED
        self.__FinalLock = 0

        # END ADDITION
```

Changes to `CheckIfPlayerHasLost`

```
    def __CheckIfPlayerHasLost(self):
        # CODE ADDED
        if self.__FinalLock == 1:
            self.__FinalLock = 2
        elif self.__Final
            print("You ved the final lock.  Your final
        # ION
        el f.__Deck.GetNumberOfCards() == 0:
```

Changes to `GetCardFromDeck`

```
                self.__MoveCard(self.__Deck, self.__Hand, self.
        # CHANGE
        if (self.__Deck.GetNumberOfCards() == 0 and self.__Hand
        self.__FinalLock < 1) or self.__Hand.GetNumberOfCards()
        # END CHANGE
            self.__GameOver = True
```

Changes to `ProcessLockSolved`

```
        print("Lock has been solved.  Your score is now:", self
        # CHANGE
        if self.__FinalLock < 2:
          while self.__Discard.Get ards() > 0:
              self.__MoveCa F Discard, self.__Deck,
              self.__Di GetCardNumberAt(0))
          self. urrle()
          t 0
          attempts < 10:
            self.__CurrentLock = self.__GetRandomLock()
            if self.__CurrentLock.IsSoluble(self.__Deck, self.__
              break
            else:
              attempts += 1
```

```
        if attempts == 10:
          print("Final Lock")
          self.__CurrentLock = self.__GenerateSolubleLock()
          self.__FinalLock = 1
          self.__GameOver = True
        # END CHANGE
```

Code for GenerateSolubleLock

```
    def __GenerateSolubleLock(self):
      cardsLeft = []
      for i in range(self.__Deck.GetNumberOfCards()):
        cardsLeft.append(self.__Deck.GetCardDescriptionAt(i))
      for i in range(self.__Hand.GetNumberOfCards()):
        cardsLeft.append(self.__Hand.GetCardDescriptionAt(i))
      newLock = Lock()
      newLock.AddChallenge(self.__GenerateChallenge(cardsLeft))
      newLock.AddChallenge(self.__GenerateChallenge(cardsLeft))
      return newLock
```

Code for GenerateChallenge

```
    def __GenerateChallenge(self, cards):
      challenge = [random.choice(cards)]
      cards.remove(challenge[-1])
      try:
        for i in range(random.randint(0,2)):
          card = random.choice(cards)
          while card[0] == challenge[-1][0]:
            card = random.choice(cards)
          challenge.append(card)
          cards.remove(card)
      except:
        pass
        # ran out of cards so return what we have
      return challenge
```

Code for IsSoluble

```
    # CODE ADDED
    def IsSoluble(self, deck, hand):
      cardsLeft = []
      for i in range(deck.GetNumberOfCards()):
        cardsLeft.append(deck.GetCardDescriptionAt(i))
      for i in range(hand.GetNumberOfCards()):
        cardsLeft.append(hand.GetCardDescriptionAt(i))
      challengesLeft = []
      for i in range(self.GetNumberOfChallenges()):
        challengesLeft.extend(self._Challenges[i].GetCondition()
      try:
        for card in challengesLeft:
          cardsLeft.remove(card)
        return True
      except:
        return False
    # END ADDITION
```

Testing:

- Printing out the final lock [1 mark] ↓

```
HAND:

-----------------------------
| P b | K a | P b | P a | K a |
-----------------------------


(D)iscard inspect, (U)se card:>
Enter a number between 1 [illegible] specify card to use:> 2
(D)iscard or (P)la[illegible]

Current s[illegible]

CURRENT LOCK
------------
Challenge met: P a, F c
Not met:       P c, K a, P b


SEQUENCE:

-------------------------------------------------------
| P a | K c | P c | F c | P c | K b | P a | F a | P a | K b |
-------------------------------------------------------
| P a | F c | P c | K a |
-----------------------


HAND:

-----------------------------
| P b | P b | P a | K [illegible]
-----------------------------

(D)iscard inspect, (U)se card:> u
Enter a number between 1 and 5 to specify card to use:> 1
(D)iscard or (P)lay?:> p

A challenge on the lock has been met.

Lock has been solved.  Your score is now: 90
You have solved the final lock.  Your final score is: 90
```

Name	

ZigZag Education supporting

A Level AQA Computer Science Paper 1

Summer 2022

BREAKTHROUGH!

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:

Programming Theory Question

Answer all questions. Remember to save this document

Q		Answer
1	(a)	
	(b)	
2	(a)	
	(b)	
	(c)	
3	(a)	
	(b)	
4	(a)	
	(b)	
5	(a)	
	(b)	
6	(a)	
	(b)	

(a)

Count	SequenceAsString	Return value
	""	
5		

(b)

Count	SequenceAsString	Return value
	""	
5		

Q		Answer
7	(a)	
	(b)	
8	(a)	
	(b)	
9	(a)	
	(b)	
	(c)	
10	(a)	
	(b)	
11		
12		
13	(a)	
	(b)	
14	(a)	
	(b)	
	(c)	
15		

Programming Tasks

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	