A-level

Computing

COMP1/Unit 1: Problem Solving, Programming, Data Representation and Practical Exercise
Mark scheme

2510
June 2013

Version: Final

Mark schemes are prepared by the Lead Assessment Writer and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation events which all associates participate in and is the scheme which was used by them in this examination. The standardisation process ensures that the mark scheme covers the students' responses to questions and that every associate understands and applies it in the same correct way. As preparation for standardisation each associate analyses a number of students' scripts: alternative answers not already covered by the mark scheme are discussed and legislated for. If, after the standardisation process, associates encounter unusual answers which have not been raised they are required to refer these to the Lead Assessment Writer.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of students' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this Mark Scheme are available from aqa.org.uk

Copyright © 2013 AQA and its licensors. All rights reserved.
AQA retains the copyright on all its publications. However, registered schools/colleges for AQA are permitted to copy material from this booklet for their own internal use, with the following important exception: AQA cannot give permission to schools/colleges to photocopy any material that is acknowledged to a third party even for internal use within the centre.

To Examiners:

1. **When to award '0' (zero) when inputting marks on QMS *and on scripts*:** A mark of 0 should be awarded where a candidate has attempted a question but failed to write anything creditworthy. Insert a hyphen when a candidate has not attempted a question. By these two actions the Principal Examiner will be able to distinguish between the two (nothing credit worthy/unattempted) when analysing any statistics.

2. This mark scheme contains the correct responses which we believe that candidates are most likely to give. Other valid responses are possible to some questions and should be credited. Examiners should refer off-mark scheme responses that they believe are creditworthy to a Team Leader.

The following annotation is used in the mark scheme:

;	- means a single mark
//	- means alternative response
/	- means an alternative word or sub-phrase
A.	- means acceptable creditworthy answer
R.	- means reject answer as not creditworthy
NE	- means not enough
I.	- means ignore
DPT	- means "Don't penalise twice". In some questions a specific error made by a candidate, if repeated, could result in the loss of more than one mark. The **DPT** label indicates that this mistake should only result in a candidate losing one mark, on the first occasion that the error is made. Provided that the answer remains understandable, subsequent marks should be awarded as if the error was not being repeated'.

No marks will be awarded for answers to testing questions where there is no evidence of programming code for the question(s) asked or where the screen captures provided by the candidate do not match what would be produced by the programming code.

Qu	Part	Marking Guidance	Marks

1	**01**	167;;
		
		If final answer is incorrect **MAX 1** can be awarded for some correct working out being shown by the candidate:
		
		1010 0111;
		10 * 16 // 160 // A * 16;
		A = 10;
		Multiplying a value by 16 and adding on 7;
			2

1	**02**	0111.1010 // 01111010 **Mark as follows:** 4 bits before binary point are 0111; 4 bits after binary point are 1010;	**2**
1	**03**	1;110 1110; **R.** if not 8 bits	**2**

1	**04**	127;	**1**

1	**05**	The number to subtract is converted into a negative number; **NE.** Convert into two's complement This is then added to the first number; Two marks for example: 23 = 00010111 -48 = 11010000; ------------- 11100111; (= -25) **A.** if not used 8 bits in examples **A.** 23 + -48 is worth 1 mark only (if there is no description) **Note:** for the first mark in the example to be awarded the two bit patterns must be correct. For the second mark in the example accept an incorrect answer as long as it is a correct addition using one of the two correct bit patterns.	**4**

1	**06**	11101110; **R.** 01110111	**1**

1	**07**	11101011; **DPT A.** 11010111	**1**

1	**08**	Get the two's complement (of a positive binary value) // Converts a positive binary value into its negative equivalent; **A.** It inverts all bits after the first 1 is received;	**1**

1	**09**	(see table below) **Mark as follows:** S0 as original state for 2nd row; 1 as output for 3rd row; Final row correct;	**3**

Input	Original State	Output	New State
0	S0	0	S0
1	S0	1	S1
0	S1	1	S1
1	S1	0	S1

2	**10**	Unicode uses more bits for each character // ASCII uses fewer bits for each character // Unicode can represent a wider range of characters // ASCII can represent a smaller range of characters // Unicode uses 16/32 bits, ASCII uses 7 bits (**A.** 8 bits);	**1**

2	11	**Role of sender:** Sender counts/checks the number of 1s in the bit pattern/value/data; and adds an extra bit to ensure even number of 1s; // Sender adds a 0 parity bit if there are an even number of 1s in the bit pattern/value/data; if odd number of 1s then a 1 parity bit is added; **Role of receiver:** Receiver counts/checks the number of 1s in the bit pattern/value/data received; if there are an odd number of 1s it identifies that an error has occurred; **A.** if even number of 1s it accepts the data received; **A.** if even number of 1s data is assumed to be correct; **A.** if odd number of 1s it requests that the data be resent; **R.** if even number of 1s, data is correct // Receiver regenerates parity bit from data received; compares generated parity bit with received parity bit – if different it identifies that an error has occurred; **A.** if the same it accepts the data received; **A.** if the same data is assumed to be correct; **A.** if different it requests that the data be resent; **R.** if the same, data is correct **A.** an odd number of errors (in the bit pattern received) will be detected; **Marking Guidance** **R.** Implication that sender or receiver are people **MAX 2** if role of sender <u>and</u> receiver not included in answer **R.** if mark point is about bit pattern being even/odd rather than the number of 1s being even/odd	**Max 4**
2	12	Error correction (not just error detection) (for single errors); Can detect when two errors have occurred in data transmission; Reduces the need for the retransmission of data; Decreases the likelihood of an undetected error // improved error detection; **R.** Implication that the parity bits are calculated by a person	**Max 1**
2	13	0;	**1**
2	14	1, 4, 8; Note: order of answers unimportant	**1**
3	15	16 (bit); **A.** 2 <u>bytes</u>	**1**
3	16	8,800,000 // 100 * 2 * 44,000;;; // 100; 2; **A.** 16÷8; **A.** different value for the sampling resolution (16) being used in the calculation but only if matches answer to part 15 44,000; **MAX 2** if final answer incorrect	**3**

3	**17**	Because of Nyquist's theorem // Because we should sample at least double the highest frequency in the original sound; Some people can hear higher frequencies than the average (so more than double has been chosen); There is no need to sample at a higher rate as humans won't notice any difference in quality above this level // sampling at a lower rate would mean that some people would notice the lower quality of the recording // sampling at a lower rate would mean that some meaningful changes in the analogue signal could be missed; higher rate would require more, <u>unnecessary</u>, storage space;	**Max 2**
3	**18**	Compression has been used; **A.** Explanation of a particular compression method that could have been used on the recording e.g. lower sampling frequency used // lower sampling resolution used;	**1**
4	**19**	Correct variable declarations for `Guess`, `NumberOfGuesses` and `NumberToGuess`; Correct prompt "`Player One enter your chosen number:` "; Followed by `NumberToGuess` assigned value entered by the user; 1st loop has syntax allowed by the programming language and one correct condition for the termination of the loop; **A.** alternative correct logic for condition 1st loop has syntax allowed by the programming language and has 2nd correct condition for the termination of the loop; **A.** alternative correct logic for condition Correct prompt "`Not a valid choice, please enter another number: `" followed by `NumberToGuess` assigned value entered by the user – must be inside the 1st iteration structure; `Guess` and `NumberOfGuesses` initialised correctly; 2nd loop has syntax allowed by the programming language and both correct conditions for the termination of the loop and is after the initialising of `Guess` and `NumberOfGuesses`; **A.** alternative correct logic for conditions Correct prompt "`Player Two have a guess: `" followed by `Guess` assigned value entered by the user – must be inside the 2nd iteration structure; `NumberOfGuesses` incremented inside the 2nd iteration structure; `If` statement with correct condition – must not be in an iterative structure; Correct output message in `Then` part of selection structure; Correct output message in `Else` part of selection structure; **I.** Case of variable names and output messages **A.** Minor typos in variable names and output messages **I.** spacing in prompts **A.** initialisation of variables at (or immediately after) declaration stage	**13**
4	**20**	****SCREEN CAPTURE**** *Must match code from 19, including prompts on screen capture matching those in code. Code for 19 must be sensible.* **Mark as follows:** 'Player One enter your chosen number: ' + user input of `0` 'Not a valid choice, please enter another number: ' Message shown;	**4**

		user input of `11` 'Not a valid choice, please enter another number: ' Message shown; user input of 5 'Player Two have a guess: ' + user input of 5; 'Player Two wins' message shown; **R.** If no evidence of user input **A.** alternative output messages if match code for 19	
4	**21**	****SCREEN CAPTURE**** *Must match code from 19, including prompts on screen capture matching those in code. Code for 19 must be sensible.* **Mark as follows:** 'Player One enter your chosen number: ' + user input of 6; 'Player Two have a guess: ' + user input of `1` 'Player Two have a guess: ' + user input of 3 'Player Two have a guess: ' + user input of 5 'Player Two have a guess: ' + user input of 7 'Player Two have a guess: ' + user input of `10`; 'Player One wins' message shown; **R.** If no evidence of user input **A.** alternative output messages if match code for 19	**3**
4	**22**	If a `FOR` loop was used then Player Two will always have 5 guesses // a `WHILE` loop will mean that the loop will terminate when Player Two guesses correctly // the number of times to iterate is not known before the loop starts;	**1**
5	**23**	`AmountToShift` // `StartPosition` // `EndPosition` // `SizeOfRailFence` // `N` // `Count` // `Key` // `ASCIICode` // `NewASCIICode` // `Count2` // `Count1` // `NoOfColumns` // `NoOfRows` // `NoOfCiphertextCharacters` // `NoOfCiphertextCharactersProcessed` // `i` // `j` // `PositionOfNextCharacter` // `LastFullRowNo` // `AmountToReduceNoOfColumnsTimesjBy` // `BeginningofNextRowIndex` // `CurrentPosition`; **R.** if any additional code **R.** if spelt incorrectly **I.** case & spaces	**1**
5	**24**	`EveryNthCharacterSteganography`; **R.** if any additional code (including routine interface) **R.** if spelt incorrectly **I.** case & spaces	**1**
5	**25**	**Pascal** `Ord` // `Length`; **VB.Net** `Asc` // `Length`; **VB6** `Asc` // `Len`; **Python**	**1**

		`ord` // `len` // `int`; **Java** `int` // `length`; **R.** if any additional code **R.** if spelt incorrectly **I.** case & spaces	

5	**26**	**Pascal**
		`Ciphertext := ''` //
		`Plaintext := ''` //
		`ChangedText := ''` //
		`TextFromFile := ''` //
		`HiddenMessage := ''`;
		I. semicolons
		
		VB.Net / VB6
		`Ciphertext = ""` //
		`Plaintext = ""` //
		`ChangedText = ""` //
		`TextFromFile = ""` //
		`HiddenMessage = ""`;
		
		Python
		`Ciphertext = ''` //
		`Plaintext = ''` //
		`ChangedText = ''` //
		`TextFromFile = ''` //
		`HiddenMessage = ''`
		
		Java
		`ciphertext = ""` //
		`plaintext = ""` //
		`changedText = ""` //
		`textFromFile = ""` //
		`hiddenMessage = ""`
		I. semicolons
		
		R. if any additional code
		R. if spelt incorrectly
		I. case & spaces
			1

5	**27**	Because if decrypt has been selected; then the plaintext alphabet needs to be shifted in the opposite direction;	**2**
5	**28**	**Mark as follows:** Identify the problem that will occur; Explanation of how `MOD 26` solves the problem; **MAX 1** if no example used in explanation **Example answer** Without `MOD 26` then the shift will only be applied correctly to letters early in the alphabet e.g. if the `AmountToShift` is 1 then the letter Z will be given a `NewASCIICode` of 91 (ASCII code for Z is 90)) and this does not	**2**

		represent a letter; Using `MOD 26` ensures that the ciphertext alphabet wraps round to the beginning of the alphabet (in this example `NewASCIICode` would become 65 the ASCII Code for A);	
5	**29**	`ApplyShiftToASCIICodeForCharacter`; **R.** if spelt incorrectly **I.** case & spaces	**1**
5	**30**	`NewASCIICode`; **A.** `ApplyShiftToASCIICodeForCharacter` (Pascal / VB.Net / VB6 only); **R.** if spelt incorrectly **I.** case & spaces	**1**

5	**31**	`GetTypeOfCharacter` //
		`Ord` (Pascal / Python only) //
		`Asc` (VB only) //
		`int` (Java only);
		
		R. if spelt incorrectly
		I. case & spaces
			1

5	**32**	**Pascal / VB6** `For 1 To Length(OriginalText)`; **VB.Net** `For 0 To (OriginalText.Length - 1)`; **Python 2/3** `for in range (0, len(OriginalText)):`; **Java** `for (count = 0; count < originalText.length(); count++)`; **A.** Alternative correct logic **A.** Any clear description that conveys correct logic	**1**
6	**33**	`If` statement with correct condition for `Key` being 0 or less; and `If` statement with correct condition for `Key` being 26 or more // one `If` statement correct condition for `Key` being 0 or less; with condition for `Key` being 26 or more linked using `Or` // one `If` statement with a list/range of values accepted between 1; and 25 (inclusive); <u>Correct</u> error message "Invalid key – a default value has been used instead" displayed and `Key` assigned a value of 1; Error message and (their) default value assigned for all, and only under all, correct conditions and the new code has been put in correct place in subroutine and value of `Key` is always returned to calling routine; **A.** Minor typos in error message **I.** Capitalisation and spacing in error message	**4**

6	**34**	****SCREEN CAPTURE**** *Must match code from 33, including prompts on screen capture matching those in code. Code for 33 must be sensible.* **Mark as follows:** Plaintext set to be `Zebra` and *option g* (encrypt) selected on menu; **I.** Case of plaintext Test showing correct working for test where `0` entered for value of `Key` // Test showing correct working for test where `27` entered for value of `Key`; Tests show correct working when values of `0, 27` and 4 entered for `Key`; **A.** Any error message as long as it matches code for 33 **R.** If case of ciphertext does not match case of plaintext **A.** alternative values for ciphertext if they would be produced by correct code from 33 based on an incorrect value for the plaintext entered by the user **Note:** Tests with value of `0` and `27` should result in error message being displayed and ciphertext of `Afcsb`. Test with value of 4 should result in no error message being shown and ciphertext of `Difve`.	**3**

7	**35**	New option "`  m.  Brute force rail fence solver`" displayed on the menu; **A.** Minor typos in menu option **I.** Capitalisation and spacing in menu option **A.** New option added anywhere in the menu **R.** if no evidence of where code has been put in the Skeleton Program (mark can be awarded if evidence can be found in either part 35 or from screen captures of test evidence in part 37)	**1**
7	**36**	New case statement for *option m*; **R.** if no evidence of where code has been put in the Skeleton Program (mark can be awarded if evidence can be found in either part 36 or from screen captures of test evidence in part 37) **A.** any character instead of m (except those used for other menu options already) – only if matches prompt from 35 `For` loop going from 2 to (length of `Ciphertext` -1); **A.** 1 instead of 2 **A.** Length of `Ciphertext` (i.e. no -1) **A.** While/Repeat loop with correct increment assignment statement Correct call to `DecryptUsingRailFence` inside repetition structure; Call to `DisplayPlaintext` inside repetition structure; `DisplayPlaintext` uses the value returned by `DecryptUsingRailFence` as a parameter;	**5**

7	**37**	****SCREEN CAPTURE****
		Must match code from 36, including prompts on screen capture matching those in code. Code for 36 must be sensible.
		
		Mark as follows:
		Entering correct ciphertext - `RLNAFCIEE`;
		New menu option displayed and *option m* entered and accepted;
		A. other character instead of m - as long as matches code for 36
		Displays correct possible plaintext messages;
		I. case of plaintext
		
		The correct plaintext messages are:
		`RCLINEAEF`
		`RAILFENCE`
		`RACELFIEN`
		`RNFIELACE`
		`RNFIEELAC`
		`RNFCIEELA`
		`RNAFCIEEL`
		
		If the repetition structure starts at 1, or goes up to length of `Ciphertext`, then 1 (or 2 if they have done both of these) `RLNAFCIEE` will also be shown (i.e. the original `Ciphertext` entered by the user).
			3

8	**38**	New option " `o. Fibonacci sequence (text from file)`" displayed on the menu; **A.** Minor typos in menu option **I.** Capitalisation and spacing in menu option **A.** New option added anywhere in the menu **R.** if no evidence of where code has been put in the Skeleton Program (mark can be awarded if evidence can be found in either part 38 or from screen captures of test evidence in part 41)	**1**
8	**39**	Created a new subroutine named `GetNthFibonacciNumber`; **A.** either a procedure or function Routine interface syntactically correct with parameter n of correct data type; **A.** parameter data type and/or return value data type unspecified (VB6 / VB.Net) Value calculated by subroutine is returned to calling routine; **R.** use of global variable Calculates value of 1 if n = 1; Calculates value of 1 if n = 2; Calculates value of 2 if n = 3 and calculates value of 3 if n = 4; Calculates value of 5 if n = 5, calculates value of 8 if n = 6 and calculates value of 13 if n = 7; 3 marks are available if there is an attempt to get the subroutine working for any positive integer n: Appropriate iterative/recursive structure; Adding $(n\text{-}1)^{th}$ and $(n\text{-}2)^{th}$ Fibonacci numbers; Calculates correct Fibonacci number for any positive integer value n and	**10**

		this value is returned to the calling routine;	
8	**40**	Additional case for *option o* added correctly; **A.** any character instead of o (except those used for other menu options already) – only if matches prompt from 38 **R.** if no evidence of where code has been put in the Skeleton Program (mark can be awarded if evidence can be found in either part 40 or from screen captures of test evidence in part 41) `Plaintext`, `N`, `StartPosition` and `EndPosition` given initial values correctly before the loop; Iterative structure has syntax allowed by the programming language and correct condition; **A**. alternative logic for condition as long as results in exactly the same working `Plaintext` changed correctly – inside the loop; **A.** answers which use a pass by reference parameter with a procedure for `GetNthFibonacciNumber` instead of a function return value Values for `N` and `StartPosition` changed correctly – inside the loop; `Plaintext` then displayed after the loop;	**6**
8	**41**	****SCREEN CAPTURE**** *Must match code from 39 and 40 including prompts on screen capture matching those in code. Code for 39 and 40 must be sensible.* **Mark as follows:** Entering requested start position (665) and end position (697) values and *option o* entered and accepted; **A.** Other character instead of o - as long as matches code for 40 Displays correct hidden message – "HelP Me"; **R.** if spelt incorrectly **R.** if capitalisation not correct	**2**

Pascal

4 **19**

```pascal
Program Question4;
  Var
    NumberToGuess : Integer;
    NumberOfGuesses : Integer;
    Guess : Integer;
  Begin
    Write('Player One enter your chosen number: ');
    Readln(NumberToGuess);
    While (NumberToGuess < 1) Or (NumberToGuess > 10)
Do
      Begin
        Write('Not a valid choice, please enter
another number: ');
        Readln(NumberToGuess);
      End;
    Guess := 0;
    NumberOfGuesses := 0;
    While (Guess <> NumberToGuess) And
(NumberOfGuesses < 5) Do
      Begin
        Write('Player Two have a guess: ');
        Readln(Guess);
        NumberOfGuesses := NumberOfGuesses + 1;
      End;
    If Guess = NumberToGuess
      Then Write('Player Two wins')
      Else Write('Player One wins');
    Readln;
  End.
```

13

6 **33**

```pascal
...
Readln(Key);
If (Key < 1) Or (Key > 25)
  Then
    Begin
      Writeln('Invalid key - a default value has been
used instead');
      Key := 1;
    End;
...
```

Alternative answer

```pascal
...
Readln(Key);
If (Key <= 0) Or (Key >= 26)
  Then
    Begin
      Writeln('Invalid key - a default value has been
used instead');
      Key := 1;
    End;
...
```

4

Alternative answer

```
Readln(Key);
If Key < 1
  Then
    Begin
      Writeln('Invalid key - a default value has been
used instead');
      Key := 1;
    End;
If Key > 25
  Then
    Begin
      Writeln('Invalid key - a default value has been
used instead');
      Key := 1;
    End;
...
```

Alternative answer

```
If Not (Key In[1..25]) Then...
```

7 | 35 — Marks: **1**

```
...
Writeln('DECRYPT');
Writeln('  j.  Caesar cipher');
Writeln('  k.  Rail fence');
Writeln('  m.  Brute force rail fence solver');
Writeln('STEGANOGRAPHY');
...
```

7 | 36 — Marks: **5**

```
...
        DisplayPlaintext(Plaintext);
      End;
'm' : Begin
        For SizeOfRailFence := 2 To
Length(Ciphertext)-1
          Do
            Begin
              Plaintext := DecryptUsingRailFence
(CipherText, SizeOfRailFence);
              DisplayPlaintext(Plaintext);
            End;
      End;
...
```

8 | 38 — Marks: **1**

```
...
Writeln('STEGANOGRAPHY');
Writeln('  n.  nth character (text from file)');
Writeln('  o.  Fibonacci sequence (text from file)');
...
```

8 **39**

Possible 10 mark answer

```
Function GetNthFibonacciNumber(N : Integer) : Integer;
  Var
    Count : Integer;
    NthFibonacciNumber : Integer;
    NMinus1thFibonacciNumber : Integer;
    NMinus2thFibonacciNumber : Integer;
  Begin
    NthFibonacciNumber := 1;
    NMinus1thFibonacciNumber := 1;
    If N > 2
      Then
        For Count := 3 To N
          Do
            Begin
              NMinus2thFibonacciNumber :=
NMinus1thFibonacciNumber;
              NMinus1thFibonacciNumber :=
NthFibonacciNumber;
              NthFibonacciNumber :=
NMinus1thFibonacciNumber + NMinus2thFibonacciNumber;
            End;
    GetNthFibonacciNumber := NthFibonacciNumber;
  End;
```

Alternative 10 mark answer

```
Function GetNthFibonacciNumber(N : Integer) : Integer;
  Var
    NthFibonacciNumber : Integer;
  Begin
    If N <= 2
      Then NthFibonacciNumber := 1
      Else NthFibonacciNumber :=
GetNthFibonacciNumber(N - 1) + GetNthFibonacciNumber(N
- 2);
    GetNthFibonacciNumber := NthFibonacciNumber;
  End;
```

10

Example answer worth 7 marks

```
Function GetNthFibonacciNumber(N : Integer) : Integer;
  Var
    NthFibonacciNumber : Integer;
  Begin
    Case N Of
      1 : NthFibonacciNumber := 1;
      2 : NthFibonacciNumber := 1;
      3 : NthFibonacciNumber := 2;
      4 : NthFibonacciNumber := 3;
      5 : NthFibonacciNumber := 5;
      6 : NthFibonacciNumber := 8;
      7 : NthFibonacciNumber := 13;
    End;
    GetNthFibonacciNumber := NthFibonacciNumber;
  End;
```

8 **40**

```
        DisplayPlaintext(Plaintext);
      End;
'o' : Begin
        Plaintext := '';
        N := 2;
        GetPositionsToUse(StartPosition, EndPosition);
        While StartPosition <= EndPosition
          Do
            Begin
              Plaintext := Plaintext +
GetTextFromFile(StartPosition, StartPosition);
              StartPosition := StartPosition +
GetNthFibonacciNumber(N);
              N := N + 1;
            End;
        DisplayPlaintext(Plaintext);
      End;
```

6

VB.Net

4 **19**

```
Module Module1
  Sub Main()
    Dim NumberToGuess As Integer
    Dim NumberOfGuesses As Integer
    Dim Guess As Integer
    Console.Write("Player One enter your chosen
number: ")
    NumberToGuess = Console.ReadLine()
    While NumberToGuess < 1 Or NumberToGuess > 10
      Console.Write("Not a valid choice, please enter
another number: ")
      NumberToGuess = Console.ReadLine()
    End While
    Guess = 0
    NumberOfGuesses = 0
    While Guess <> NumberToGuess And NumberOfGuesses <
5
      Console.Write("Player Two have a guess: ")
      Guess = Console.ReadLine()
      NumberOfGuesses = NumberOfGuesses + 1
    End While
    If Guess = NumberToGuess Then
      Console.Write("Player Two wins")
    Else
      Console.Write("Player One wins")
    End If
    Console.ReadLine()
  End Sub
End Module
```

13

6 **33**

```
...
Key = Console.ReadLine
If Key < 1 Or Key > 25 Then
  Console.WriteLine("Invalid key – a default value has
been used instead")
  Key = 1
End If
...
```

Alternative answer

```
...
Key = Console.ReadLine
If Key <= 0 Or Key >= 26 Then
  Console.WriteLine("Invalid key – a default value has
been used instead")
  Key = 1
End If
...
```

4

Alternative answer

```
...
Key = Console.ReadLine
If Key < 1 Then
  Console.WriteLine("Invalid key – a default value has
been used instead")
  Key = 1
End If
If Key > 25 Then
  Console.WriteLine("Invalid key – a default value has
been used instead")
  Key = 1
End If

...
```

7 **35** — **1**

```
Sub DisplayMenu()
  ...
  Console.WriteLine("DECRYPT")
  Console.WriteLine("  j.  Caesar cipher")
  Console.WriteLine("  k.  Rail fence cipher")
  Console.WriteLine("  m.  Brute force rail fence
solver")
  Console.WriteLine("STEGANOGRAPHY")
  ...
End Sub
```

7 **36** — **5**

```
...
Case "m"
  For SizeOfRailFence = 2 To (Ciphertext.Length - 1)
    Plaintext = DecryptUsingRailFence(Ciphertext,
SizeOfRailFence)
    DisplayPlaintext(Plaintext)
  Next
...
```

8 **38** — **1**

```
Sub DisplayMenu()
  ...
  Console.WriteLine("STEGANOGRAPHY")
  Console.WriteLine("  n.  nth character (text from
file)")
  Console.WriteLine("  o.  Fibonacci sequence (text
from file)”)
  ...
End Sub
```

8 **39** — **10**

Possible 10 mark answer

```
Function GetNthFibonacciNumber(ByVal N As Integer) As
Integer
  Dim Count As Integer
  Dim NthFibonacciNumber As Integer
  Dim NMinus1thFibonacciNumber As Integer
  Dim NMinus2thFibonacciNumber As Integer
  NthFibonacciNumber = 1
  NMinus1thFibonacciNumber = 1
  For Count = 3 To N
    NMinus2thFibonacciNumber =
NMinus1thFibonacciNumber
    NMinus1thFibonacciNumber = NthFibonacciNumber
    NthFibonacciNumber = NMinus1thFibonacciNumber +
NMinus2thFibonacciNumber
  Next
  GetNthFibonacciNumber = NthFibonacciNumber
End Function
```

Alternative 10 mark answer

```
Function GetNthFibonacciNumber(ByVal N As Integer) As
Integer
  Dim NthFibonacciNumber As Integer
  If N <= 2 Then
    NthFibonacciNumber = 1
  Else
    NthFibonacciNumber = GetNthFibonacciNumber(N - 1)
+ GetNthFibonacciNumber(N - 2)
  End If
  GetNthFibonacciNumber = NthFibonacciNumber
End Function
```

Example answer worth 7 marks

```
Function GetNthFibonacciNumber(ByVal N As Integer) As
Integer
  Dim NthFibonacciNumber As Integer
  Select Case N
    Case 1
      NthFibonacciNumber = 1
    Case 2
      NthFibonacciNumber = 1
    Case 3
      NthFibonacciNumber = 2
    Case 4
      NthFibonacciNumber = 3
    Case 5
      NthFibonacciNumber = 5
    Case 6
      NthFibonacciNumber = 8
    Case 7
      NthFibonacciNumber = 13
  End Select
  GetNthFibonacciNumber = NthFibonacciNumber
End Function
```

8	**40**	`...`
		`  DisplayPlainText(Plaintext)`
		`Case "o"`
		`  Plaintext = ""`
			6

```
  N = 2
  GetPositionsToUse(StartPosition, EndPosition)
  While StartPosition <= EndPosition
    Plaintext = Plaintext &
GetTextFromFile(StartPosition, StartPosition)
    StartPosition = StartPosition +
GetNthFibonacciNumber(N)
    N = N + 1
  End While
  DisplayPlainText(Plaintext)
End Select
...
```

VB6

4 **19**

```
Private Sub Form_Load()
  Dim NumberToGuess As Integer
  Dim NumberOfGuesses As Integer
  Dim Guess As Integer
  NumberToGuess = ReadLine("Player One enter your
chosen number: ")
  While NumberToGuess < 1 Or NumberToGuess > 10
    NumberToGuess = ReadLine("Not a valid choice,
please enter another number: ")
  Wend
  Guess = 0
  NumberOfGuesses = 0
  While Guess <> NumberToGuess And NumberOfGuesses < 5
    Guess = ReadLine("Player Two have a guess: ")
    NumberOfGuesses = NumberOfGuesses + 1
  Wend
  If Guess = NumberToGuess Then
    WriteLineWithMsg ("Player Two wins")
  Else
    WriteLineWithMsg ("Player One wins")
  End If
End Sub
```

Alternative answers could use some of the following instead of WriteLineWithMsg / ReadLine:

```
Text1.Text = Text1.Text & ...
WriteLine
WriteWithMsg
Msgbox
InputBox
WriteNoLine
```

13

6 **33**

```
...
Key = ReadLine("Enter the amount that shifts the plain
alphabet to the cipher alphabet: ")
If Key < 1 Or Key > 25 Then
  WriteLineWithMsg ("Invalid key – a default value has
been used instead")
  Key = 1
End If
...
```

A. `Text1.Text = Text1.Text & "Invalid key – a default value has been used instead"`

A. `WriteWithMsg ("Invalid key – a default value has been used instead")`

A. `Msgbox ("Invalid key – a default value has been used instead")`

A. `WriteNoLine ("Invalid key – a default value has been used instead")`

Alternative answer

4

```
...
Key = ReadLine("Enter the amount that shifts the plain
alphabet to the cipher alphabet: ")
If Key <= 0 Or Key >= 26 Then
  WriteLineWithMsg ("Invalid key – a default value has
been used instead")
  Key = 1
End If
...
```

Alternative answer

```
...
Key = Console.ReadLine
If Key < 1 Then
  WriteLineWithMsg ("Invalid key – a default value has
been used instead")
  Key = 1
End If
If Key > 25 Then
  WriteLineWithMsg ("Invalid key – a default value has
been used instead")
  Key = 1
End If
...
```

7 **35**

```
Private Sub DisplayMenu()
  ...
  WriteLine ("DECRYPT")
  WriteLine ("  j.  Caesar cipher")
  WriteLine ("  k.  Rail fence cipher")
  WriteLine ("  m.  Brute force rail fence solver")
  WriteLine ("STEGANOGRAPHY")
  ...
End Sub
```

1

7 **36**

```
...
Case "m"
  For SizeOfRailFence = 2 To (Len(Ciphertext)- 1)
    Plaintext = DecryptUsingRailFence(Ciphertext,
SizeOfRailFence)
    Call DisplayPlaintext(Plaintext)
  Next
...
```

5

8 **38**

```
Private Sub DisplayMenu()
  ...
  WriteLine ("STEGANOGRAPHY")
  WriteLine ("  n.  nth character (text from file)")
  WriteLine ("  o.  Fibonacci sequence (text from
file)")
  ...
End Sub
```

1

8	39	**Possible 10 mark answer**
		
		`Private Function GetNthFibonacciNumber(ByVal N As Integer) As Integer`
		`  Dim Count As Integer`
		`  Dim NthFibonacciNumber As Integer`
		`  Dim NMinus1thFibonacciNumber As Integer`
		`  Dim NMinus2thFibonacciNumber As Integer`
		`  NthFibonacciNumber = 1`
		`  NMinus1thFibonacciNumber = 1`
		`  For Count = 3 To N`
		`    NMinus2thFibonacciNumber = NMinus1thFibonacciNumber`
		`    NMinus1thFibonacciNumber = NthFibonacciNumber`
		`    NthFibonacciNumber = NMinus1thFibonacciNumber + NMinus2thFibonacciNumber`
		`  Next`
		`  GetNthFibonacciNumber = NthFibonacciNumber`
		`End Function`
		
		Alternative 10 mark answer
		
		`Private Function GetNthFibonacciNumber(ByVal N As Integer) As Integer`
		`  Dim NthFibonacciNumber As Integer`
		`  If N <= 2 Then`
		`    NthFibonacciNumber = 1`
		`  Else`
		`    NthFibonacciNumber = GetNthFibonacciNumber(N - 1) + GetNthFibonacciNumber(N - 2)`
		`  End If`
		`  GetNthFibonacciNumber = NthFibonacciNumber`
		`End Function`
		
		Example answer worth 7 marks
		
		`Private Function GetNthFibonacciNumber(ByVal N As Integer) As Integer`
		`  Dim NthFibonacciNumber As Integer`
		`  Select Case N`
		`    Case 1`
		`      NthFibonacciNumber = 1`
		`    Case 2`
		`      NthFibonacciNumber = 1`
		`    Case 3`
		`      NthFibonacciNumber = 2`
		`    Case 4`
		`      NthFibonacciNumber = 3`
		`    Case 5`
		`      NthFibonacciNumber = 5`
		`    Case 6`
		`      NthFibonacciNumber = 8`
		`    Case 7`
		`      NthFibonacciNumber = 13`
		`  End Select`
			10

		GetNthFibonacciNumber = NthFibonacciNumber End Function	

8	**40**	...
		  Call DisplayPlaintext(Plaintext)
		Case "o"
		  Plaintext = ""
		  N = 2
		  Call GetPositionsToUse(StartPosition, EndPosition)
		  While StartPosition <= EndPosition
		    Plaintext = Plaintext &
		GetTextFromFile(StartPosition, StartPosition)
		    StartPosition = StartPosition +
		GetNthFibonacciNumber(N)
		    N = N + 1
		  Wend
		  Call DisplayPlaintext(Plaintext)
		End Select
		...
			6

Python 3

4	**19**	`#  Question 4`
		`print('Player One enter your chosen number: ')`
		`NumberToGuess = int(input())`
		`while (NumberToGuess < 1) or (NumberToGuess > 10) :`
		  `print('Not a valid choice, please enter another number: ')`
		  `NumberToGuess = int(input())`
		`Guess = 0`
		`NumberOfGuesses = 0`
		`while (Guess != NumberToGuess) and (NumberOfGuesses < 5) :`
		  `print('Player Two have a guess: ')`
		  `Guess = int(input())`
		  `NumberOfGuesses = NumberOfGuesses + 1`
		`if Guess == NumberToGuess :`
		  `print('Player Two wins')`
		`else :`
		  `print('Player One wins')`
		
		Alternative print/input combinations:
		
		`NumberToGuess = int(input('Player One enter your chosen number: '))`
		
		`Guess = int(input('Player Two have a guess: '))`
			13

6	**33**	`. . .`
		`Key = int(input('Enter the amount that shifts the plain alphabet to the cipher alphabet: '))`
		`if (Key < 1) or (Key > 25):`
		  `print('Invalid key – a default value has been used instead')`
		  `Key = 1`
		`. . .`
		
		Alternative answer:
		
		`. . .`
		`Key = int(input('Enter the amount that shifts the plain alphabet to the cipher alphabet: '))`
		`if (Key <= 0) or (Key >= 26):`
		  `print('Invalid key – a default value has been used instead')`
		  `Key = 1`
		`. . .`
		
		Alternative answer:
		
		`. . .`
		`Key = int(input('Enter the amount that shifts the plain alphabet to the cipher alphabet: '))`
		`if Key < 1:`
			4

		** print('Invalid key – a default value has been used instead')** ** Key = 1** **if Key > 25:** ** print('Invalid key – a default value has been used instead')** ** Key = 1** . . . Alternative Answer **if not Key in [1,25]:** . . .	

7	35	. . .
		print('DECRYPT')
		print(' j. Caesar cipher')
		print(' k. Rail fence')
		print(' m. Brute force rail fence solver')
		print('STEGANOGRAPHY')
		. . .
			1

7	36	. . . DisplayPlaintext(Plaintext) **elif Choice == 'm':** ** for SizeOfRailFence in range (2, len(CipherText):** ** Plaintext = DecryptUsingRailFence(CipherText, SizeOfRailFence)** ** DisplayPlaintext(Plaintext**) . . .	5

8	38	def DisplayMenu():
		. . .
		  print('STEGANOGRAPHY')
		  print(' n. nth character (text from file) ')
		** print(' o. Fibonacci sequence (text from file) ')**
		. . .
			1

8	39	**Possible 10 mark answer**
		
		def GetFibonacciNumber(N):
		  NthFibonacciNumber = 1
		  NMinus1thFibonacciNumber = 1
		  for Count in range (3, N + 1):
		    NMinus2thFibonacciNumber = NMinus1thFibonacciNumber
		    NMinus1thFibonacciNumber = NthFibonacciNumber
		    NthFibonacciNumber = NMinus1thFibonacciNumber + NMinus2thFibonacciNumber
		  return NthFibonacciNumber
		
		Alternative 10 mark answer
		
		def GetNthFibonacciNumber(N):
			10

```
  if N <= 2:
    NthFibonacciNumber = 1
  else:
    NthFibonacciNumber = GetNthFibonacciNumber(N – 1)
+ GetNthFibonacciNumber(N – 2)
  return NthFibonacciNumber
```

Example answer worth 7 marks

```
def GetNthFibonacciNumber(N):
  if N == 1:
    NthFibonacciNumber = 1
  elif N == 2:
    NthFibonacciNumber = 1
  elif N == 3:
    NthFibonacciNumber = 2
  elif N == 4:
    NthFibonacciNumber = 3
  elif N == 5:
    NthFibonacciNumber = 5
  elif N == 6:
    NthFibonacciNumber = 8
  elif N == 7:
    NthFibonacciNumber = 13
  return NthFibonacciNumber
```

8 **40**

```
. . .
elif Choice == 'o' :
  Plaintext = ""
  N = 2
  StartPosition, EndPosition = GetPositionsToUse()
  while StartPosition <= EndPosition :
    Plaintext = Plaintext +
GetTextFromFile(StartPosition, StartPosition)
    StartPosition = StartPosition +
GetNthFibonacciNumber(N)
     N = N + 1
  DisplayPlaintext(Plaintext)
. . .
```

6

Python 2

4 | **19** | **13**

```
#  Question 4
print 'Player One enter your chosen number: '
NumberToGuess = int(raw_input())
while (NumberToGuess < 1) or (NumberToGuess > 10) :
  print 'Not a valid choice, please enter another
number: '
  NumberToGuess = int(raw_input())
Guess = 0
NumberOfGuesses = 0
while (Guess != NumberToGuess) and (NumberOfGuesses <
5) :
  print 'Player Two have a guess: '
  Guess = int(raw_input())
  NumberOfGuesses = NumberOfGuesses + 1
if Guess == NumberToGuess :
  print 'Player Two wins'
else :
  print 'Player One wins'
```

Alternative print/input combinations:

```
NumberToGuess = int(raw_input('Player One enter your
chosen number: '))

Guess = int(raw_input('Player Two have a guess: '))
```

6 | **33** | **4**

```
Key = int(raw_input('Enter the amount that shifts the
plain alphabet to the cipher alphabet: '))
if (Key < 1) or (Key > 25):
  print 'Invalid key – a default value has been used
instead'
  Key = 1
. . .
```

Alternative answer:

```
. . .
Key = int(raw_input('Enter the amount that shifts the
plain alphabet to the cipher alphabet: '))
if (Key <= 0) or (Key >= 26):
  print 'Invalid key – a default value has been used
instead'
  Key = 1
. . .
```

Alternative answer:

```
. . .
Key = int(raw_input('Enter the amount that shifts the
plain alphabet to the cipher alphabet: '))
if Key < 1:
  print 'Invalid key – a default value has been used
instead'
```

```
  Key = 1
if Key > 25:
  print 'Invalid key – a default value has been used
instead'
  Key = 1
. . .
```

Alternative Answer

```
if not Key in [1,25]:
. . .
```

7 **35**

```
. . .
print 'DECRYPT'
print '  j.  Caesar cipher'
print '  k.  Rail fence'
print '  m.  Brute force rail fence solver'
print 'STEGANOGRAPHY'
. . .
```

1

7 **36**

```
. . .
  DisplayPlaintext(Plaintext)
elif Choice == 'm':
  for SizeOfRailFence in range (2, len(CipherText):
    Plaintext = DecryptUsingRailFence(CipherText,
SizeOfRailFence)
    DisplayPlaintext(Plaintext)
. . .
```

5

8 **38**

```
def DisplayMenu():
. . .
    print 'STEGANOGRAPHY'
    print '  n.  nth character (text from file) '
    print '  o.  Fibonacci sequence (text from file) '
. . .
```

1

8 **39**

Possible 10 mark answer

```
def GetFibonacciNumber(N):
  NthFibonacciNumber = 1
  NMinus1thFibonacciNumber = 1
  for Count in range (3, N + 1):
    NMinus2thFibonacciNumber =
NMinus1thFibonacciNumber
    NMinus1thFibonacciNumber = NthFibonacciNumber
    NthFibonacciNumber = NMinus1thFibonacciNumber +
NMinus2thFibonacciNumber
  return NthFibonacciNumber
```

Alternative 10 mark answer

```
def GetNthFibonacciNumber(N):
```

10

```
  if N <= 2:
    NthFibonacciNumber = 1
  else:
    NthFibonacciNumber = GetNthFibonacciNumber(N – 1)
+ GetNthFibonacciNumber(N – 2)
  return NthFibonacciNumber
```

Example answer worth 7 marks

```
def GetNthFibonacciNumber(N):
  if N == 1:
    NthFibonacciNumber = 1
  elif N == 2:
    NthFibonacciNumber = 1
  elif N == 3:
    NthFibonacciNumber = 2
  elif N == 4:
    NthFibonacciNumber = 3
  elif N == 5:
    NthFibonacciNumber = 5
  elif N == 6:
    NthFibonacciNumber = 8
  elif N == 7:
    NthFibonacciNumber = 13
  return NthFibonacciNumber
```

8 **40**

```
. . .
elif Choice == 'o' :
  Plaintext = ""
  N = 2
  StartPosition, EndPosition = GetPositionsToUse()
  while StartPosition <= EndPosition :
    Plaintext = Plaintext +
GetTextFromFile(StartPosition, StartPosition)
    StartPosition = StartPosition +
GetNthFibonacciNumber(N)
    N = N + 1
  DisplayPlaintext(Plaintext)
. . .
```

6

Java

4 **19**

```
int numberToGuess;
int numberOfGuesses;
int guess;
numberToGuess = console.readInteger("Player One enter your
chosen number: ");
while(numberToGuess < 1 || numberToGuess > 10){
  numberToGuess = console.readInteger("Not a valid choice,
please enter another number: ");
}
guess = 0;
numberOfGuesses = 0;
while (guess != numberToGuess && numberOfGuesses < 5){
  guess = console.readInteger("Player Two have a guess: ");
  numberOfGuesses++;
}
if(guess == numberToGuess){
  console.println("Player Two wins");
}else{
  console.println("Player One wins");
}
```

13

6 **33**

```
. . .
key = console.readInteger("Enter the amount that
shifts the plain alphabet to the cipher alphabet: ");
if (key < 1 || key > 25) {
  console.println("Invalid key - a default value has
been used instead.");
  key = 1;
}
. . .
```

Alternative answer:

```
. . .
key = console.readInteger("Enter the amount that
shifts the plain alphabet to the cipher alphabet: ");
if (key <= 0 || key >= 26) {
  console.println("Invalid key - a default value has
been used instead.");
  key = 1;
}
. . .
```

Alternative answer:

```
. . .
key = console.readInteger("Enter the amount that
shifts the plain alphabet to the cipher alphabet: ");
if (key < 1 ) {
  console.println("Invalid key - a default value has
been used instead.");
  key = 1;
}
if (key > 25 ) {
```

4

		console.println("Invalid key - a default value has been used instead.");
		  key = 1;
		}
		. . .	

7	**35**	. . .
		console.println("DECRYPT");
		console.println(" j. Caesar cipher");
		console.println(" k. Rail fence");
		console.println(" m. Brute force rail fence solver");
		console.println("STEGANOGRAPHY");. . .
		. . .
			1

7	**36**	. . .
		  displayPlaintext(plaintext);
		  break;
		}
		case 'm': {
		  for(sizeOfRailFence = 2; sizeOfRailFence < cipherText.length(); sizeOfRailFence++) {
		    plainText = decryptUsingRailFence(cipherText, sizeOfRailFence);
		    displayPlaintext(plaintext);
		  }
		  break;
		}
		. . .
			5

8	**38**	void displayMenu() {
		. . .
		  console.println("STEGANOGRAPHY");
		  console.println(" n. nth character (text from file)");
		  console.println(" o. Fibonacci sequence (text from file)");
		. . .
			1

8	**39**	**Possible 10 mark answer**
		
		int getNthFibonacciNumber(int n) {
		  int count;
		  int nthFibonacciNumber;
		  int nMinus1thFibonacciNumber;
		  int nMinus2thFibonacciNumber;
		  nthFibonacciNumber = 1;
		  nMinus1thFibonacciNumber = 1;
		  for(count = 3; count <= n; count++) {
		    nMinus2theFibonacciNumber = nMinus1thFibonacciNumber;
		    nMinus1thFibonacciNumber = nthFibonacciNumber;
		    nthFibonacciNumber = nMinus1thFibonacciNumber +
			10

```
nMinus2thFibonacciNumber;
  }
  return nthFibonacciNumber;
}
```

Alternative 10 mark answer

```
int getNthFibonacciNumber(int n) {
  int nthFibonacciNumber;
  if (n <= 2) {
    nthFibonacciNumber = 1;
  } else {
    nthFibonacciNumber = getNthFibonacciNumber(n – 1)
+ getNthFibonacciNumber(n – 2);
  }
  return nthFibonacciNumber;
}
```

Example answer worth 7 marks

```
int getNthFibonacciNumber(int n) {
  int nthFibonacciNumber;
  if (n == 1) {
    nthFibonacciNumber = 1;
  } else if (n == 2) {
    nthFibonacciNumber = 1;
  } else if (n == 3) {
    nthFibonacciNumber = 2;
  } else if (n == 4) {
    nthFibonacciNumber = 3;
  } else if (n == 5) {
    nthFibonacciNumber = 5;
  } else if (n == 6) {
    nthFibonacciNumber = 8;
  } else if (n == 7) {
    nthFibonacciNumber = 13;
  }
  return nthFibonacciNumber;
}
```

8 **40**

```
. . .
case 'o': {
  plaintext = "";
  n = 2;
  getPositionsToUse(start, end);
  while(start.position <= end.position) {
    plaintext += getTextFromFile(start.position,
start.position);
    start.position += getNthFibonacciNumber(n);
    n++;
  }
  displayPlaintext(plaintext);
  break;
}
. . .
```

6

Statistical data and information on grade boundary ranges www.aqa.org.uk/over/stat.html

UMS conversion calculator www.aqa.org.uk/umsconversion