version 1.0

ASSESSMENT and QUALIFICATIONS ALLIANCE

General Certificate of Education

Computing 2510

UNIT 1 Problem Solving, Programming, Data Representation and Practical Exercise

Mark Scheme

2009 examination - June series

This mark scheme uses the new numbering system which is being introduced for examinations from June 2010

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

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of candidates' 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.

Set and published by the Assessment and Qualifications Alliance.

The Assessment and Qualifications Alliance (AQA) is a company limited by guarantee registered in England and Wales (company number 3644723) and a registered charity (registered charity number 1073334).
Registered address: AQA, Devas Street, Manchester M15 6EX *Dr Michael Cresswell Director General*

Notation used mark schemes:

; - 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
I - means ignore.

Question 1			
0	**1**	Smallest ; picture element // unit which can be drawn on screen // addressable/resolvable part/unit of a picture ;	**2**
0	**2**	<table><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr></table>	**1**
0	**3**	184	**1**
0	**4**	pixels are stored as numbers // bit patterns/binary code //RGB bits	**1**
0	**5**	8 **; A.** 1 byte	**1**
0	**6**	drawing is made up of drawing <u>objects</u> // or by example e.g. drawing is made up of circle/rectangle/straight line/etc. (must give at least two example objects) ; different objects(**A.** shapes) have a defined set of <u>properties</u> // or by example ; objects are stored as drawing commands/drawing list ; some properties use mathematical equations/formulae ;	**Max 2**
0	**7**	object type ; co-ordinates/location of the <u>centre</u> **R.** centre (only) ; radius/diameter ; fill colour ; fill style ; line thickness ; line colour ; line style ; anything reasonable ; **R.** colour (only) Position (only)	**Max 3**

Question 2			
0	**8**	Any 8-bit pattern padded with 0's ; ending in 1011 ;	**2**
0	**9**	<u>0</u>111 0111 ;	**1**
1	**0**	1 ; 111 0011 ;	**2**
1	**1**	seven leading 0's ; 11001 ; (.) 1100 ;	**3**

Question 3			
1	2	stationary at floor X / stationary at floor Y (MAX 2) ; moving (or equivalent) up ; moving down ; full ; out of order ; emergency stopped; door open ; door closed ; lift has been requested for floor X (MAX 1) ; **A.** 'stationary' for 1 (but not in addition to the above) **A.** 'moving' alone for 1 (but not in addition to the above) A. anything plausible R. anything which reads like an action/input	**Max 3**
1	3	S1 ;	**1**
1	4	S1 ;	**1**
1	5	S2 ;	**1**

Question 4			
1	6	**** SCREEN CAPTURE **** “The new word?” + setter input 'EAGLE' ; input of correct guess 'EAGLE' ; (**A.** ‘eagle’ if code in (b) has evidence for use of function Ucase, .ToUpper, etc.) correct logic demonstrated with “CORRECT” ; **NB** VB6 – all three stages must be evidenced	**3**
1	7	**** SCREEN CAPTURE **** setter input 'BEAR' ; “Your guess?” + any incorrect guess ; correct logic demonstrated with “INCORRECT” ; **NB** VB6 – all three stages must be evidenced	**3**

1	8	**Visual Basic**
		`Dim NewWord As String`
		`Dim UserWordGuess As String`
		`Console.Write("The new word?")`
		`NewWord = Console.ReadLine`
		
		`Console.Write("Your guess?")`
		`UserWordGuess = Console.ReadLine`
		
		`If UserWordGuess = NewWord`
		`  Then Console.WriteLine("CORRECT")`
		`  Else Console.WriteLine("INCORRECT")`	

```
End If
```

Pascal

```
Var
  NewWord : String;
  UserWordGuess : String;

Begin
  Write('The new word? ');
  Readln(NewWord);
  Write('Your guess?');
  Readln(UserWordGuess);
  If UserWordGuess = NewWord
    Then Writeln('CORRECT')
    Else Writeln('INCORRECT');

  Readln;
End.
```

Mark as follows:

evidence of two variables declared ;
data types appropriate to the language for both variables ;
correct two identifier names used - NewWord - UserWordGuess ;
(**A.** case variations)

correct user prompt "The new word?" (**A.** 'imprecise') ;

correctly formed IF followed by condition ;
THEN clause followed by the logically correct output (**A.** 'imprecise') ;
ELSE clause ; followed by the logically correct output (**A.** 'imprecise') ;

NB. Two separate IF statements scores maximum 2

Max 7

Question 5			
1	**9**	section of code can be referred to by name ; aids readability ; aids testing ; code is easier to maintain/debug ; the same block of code can be used repeatedly within the program ; reusable within other programs ; they encourage the use of local variables ; reduces the complexity/results in less code in the main body of the program ; they are 'building blocks' for structured programming ;	**Max 3**

2	**0**	**General:** Do not give credit for variables which are stated as part of an assignment statement **A.** Variable shown in a declaration statement
		`PhraseOK ; ThisNewPhrase (Java only Phrase) ; Position ;`
		`GuessedLetter ; MissingLetter ;`
		`Python only – Choice`
			Max 1

2	**1**	`NewPhrase ; PhraseHasBeenSet ; PhraseGuessed ; ;`
		`GuessStatusArray ; LettersGuessedArray ; NextGuessedLetter ;`
		`Index ; Choice (not Python)`
		
		`VB and VB6 only - IndividualLettersArray`
		`Java only – Console`
			Max 1

2	**2**	`Len / Length/ StrLen;`
		`PHP – Trim, , IntVal`
		`C# - int.Parse`
		`Python – Range`
		`Java – ReadLine – ReadChar – CharAt`
			1

2	**3**	`GuessStatusArray ; LettersGuessedArray ;` `VB.Net and VB6 only: IndividualLettersArray ;`	**1**
2	**4**	`Position ; Index ; (`**`A.`**` PhraseOK / Missingletter / Choice)` `Java only – Found – i`	**1**
2	**5**	`DisplayMenu ; DisplayCurrentStatus ;`	**1**

2	**6**	`DisplayCurrentStatus ; AllLettersGuessedCorrectly ;`
		`SetUpGuessStatusArray ;`
		
		`Java only - GetNewPhrase ;`
		`Java / Python only – HasLetterBeenUsed ;`
		`C, C#, java – main`
			1

<table>
<tr><td>2</td><td>7</td><td>Check carefully with (c) (i)

<table><tr><td>AllLettersGuessedCorrectly
(Not Python)</td><td>NewPhrase
GuessStatusArray
IndividualLettersArray(VB6 only)</td></tr><tr><td>SetUpGuessStatusArray</td><td>NewPhrase (+GuessStatusArray Java only)
GuessStatusArray (not PHP/C#)
IndividualLettersArray(VB.Net and VB6 only)</td></tr><tr><td>DisplayCurrentStatus</td><td>GuessStatusArray
Phraselength</td></tr><tr><td>GetNewPhrase (Java only)</td><td>minimumLength</td></tr><tr><td>main (C, C#, java only)</td><td>args</td></tr><tr><td>hasLetterBeenUsed
(Java and Python only)</td><td>LettersGuessedArray, myGuess</td></tr></table>(Python Letter only)</td><td>1</td></tr>
<tr><td>2</td><td>8</td><td>takes the original word/phrase (A. by implication);
checks its length using <u>characters</u>;
“a length of less than 10 is not permitted” / equivalent statement with the <u>exact logic</u>;</td><td>3</td></tr>
<tr><td>2</td><td>9</td><td>PhraseOK = True / PhraseOK = False / PhraseOK / or explained ;</td><td>1</td></tr>
<tr><td>3</td><td>0</td><td>program will continually prompt the setter for a new phrase ;
there is a continuous loop ;</td><td>Max 1</td></tr>
<tr><td>3</td><td>1</td><td>a section of code needs to be repeated // A. by implication e.g. “done for each character in the string” ;</td><td>1</td></tr>
<tr><td>3</td><td>2</td><td>the number of iterations is known // the loop is to iterate a (R. fixed) known no. of times ;</td><td>1</td></tr>
<tr><td>3</td><td>3</td><td>The number of <u>characters</u> (R. Letters) /length of the phrase ;</td><td>1</td></tr>
</table>

Question 6

3 4

Key positions are: 2; 5; 6; 10;

Index	1	2	3	4	5	6	7	8	9	10	11
		+			+	+				+	

Each correct index position ; (MAX 4)
Some 'indicator' value e.g. True or equivalent used for all correct positions ;
A. could be the actual letters stored (all in correct positions) **5**

3 5

No (change) // an attempt will be made to overwrite the existing 'F' entry at position 6 in the array ; **1**

3 6

Key positions are: 1- 2-3-4 ;

Index	1	2	3	4	5	6	7	8	9	10	11
	'C'	'G'	'B'	'H'							

First four cells used ;
and contain the correct letters ; **2**

3 7

No change // **A.** changes; **1**

3 8

No change followed by "the same letter is never stored more than once" / "the letter has already been entered" ;

A *different possible interpretation …*
Changes followed by "Second 'B' character is stored at position 5" ; **1**

Question 7

3 9

Visual Basic

```
Sub DisplayMenu()
Console.Writeline("_____________________________")
Console.WriteLine("1. SETTER - Makes new word/phrase")
Console.WriteLine("")
Console.WriteLine("2. USER - Next letter guess")
Console.WriteLine("")
Console.WriteLine("3. USER - Make a complete word/phrase guess")
Console.WriteLine("")
Console.WriteLine("5. End")
End Sub
```

		Pascal
		`Procedure DisplayMenu;`
		`  Begin`
		`  Writeln('_____');`
		`  Writeln;`
		`  Writeln('1. SETTER - Makes new word/phrase');`
		`  Writeln;`
		`  Writeln('2. USER - Next letter guess');`
		`  Writeln('');`
		`  Writeln('3. USER – Make a complete word/phrase guess');`
		`  Writeln;`
		`  Writeln('5. End');`
		`  Writeln;`
		`  End;`
		
		Mark as follows:
		additional choice for option 3 shown (**A.** minor typos) ;
		inside procedure DisplayMenu ;
		VB6 - code added to listbox control lstMenu ; inside Form_Load event ;
			2

4	**0**	**Visual Basic**
		`Sub InputUsersCompletePhraseGuess()`
		`  Console.WriteLine("Procedure InputUsersCompletePhraseGuess has`
		`  been called")`
		`  Console.ReadLine()`
		`End Sub`
		
		Pascal
		`Procedure InputUsersCompletePhraseGuess;`
		`  begin`
		`  Writeln('Procedure InputUsersCompletePhraseGuess has been called ');`
		`  end;`
		Mark as follows:
		New procedure `InputUsersCompletePhraseGuess()` defined ;
		Contains the required output (A. minor typos);
		VB6 = `MsgBox " Appropriate text …"`
		
		End of the procedure/function is clear ;
		Python only : Award 3rd mark for correct indentation of print statement.
			3

4	1	**Visual Basic**
		`If Choice = 3 Then Call InputUsersCompletePhraseGuess()`
		
		Pascal
		`If Choice = 3`
		`  Then`
		`    Begin`
		`      InputUsersCompletePhraseGuess`
		`    End;`
		
		Mark as follows:
		Call to procedure InputUsersCompletePhraseGuess ;
		IF statement for choice 3 ;
			2

4	2	**** SCREEN CAPTURE ***** Menu choice 3 selected ; 'Correct' output message displayed - Must match text in code for (b) ;	**2**

Question 8

4	3	**Visual Basic**
		`    Sub CountPhrasesFromFile()`
		`        ' uses global variable NumberOfPhrasesInFile`
		`        Dim TempPhrase As String`
		
		`        FileOpen(1, "MyPhrases.txt", OpenMode.Input)`
		`        NumberOfPhrasesInFile = 0`
		`        Do`
		`            TempPhrase = LineInput(1)`
		`            NumberOfPhrasesInFile = NumberOfPhrasesInFile + 1`
		`        Loop Until EOF(1)`
		`        FileClose(1)`
		`    End Sub`
		
		OR equivalent using the FileStream object and StreamReader method.
		
		Pascal
		answer with WHILE loop:
		`Procedure CountPhrasesFromFile;`
		`{ uses global variable NumberOfPhrasesInFile }`
		
		`Var`
		`  TempPhrase:String;`
		
		`Begin`
		`  Reset(MyPhrasesPipe);`
		`  NumberOfPhrasesInFile:=0;`
		`  While Not Eof(MyPhrasesPipe)`
		`    Do`	

```
      Begin
        ReadLn(MyPhrasesPipe, TempPhrase);
        NumberOfPhrasesInFile:=NumberOfPhrasesInFile+1;
      End;

  Close(MyPhrasesPipe);
End;
```

Alternative implementations:

```
Procedure CountPhrasesInFile(Var NumberOfPhrasesInFile :
Integer);
Function CountPhrasesInFile(Var NumberOfPhrasesInFile :
                                          Integer) : Integer;
```

Mark as follows:

open file correctly formed ;
correctly formed loop (post or pre condition);
terminates with 'EOF' ;
each phrase read from file ;
temporary variable used to store the next line of text ;
file closed ;
"NumberOfPhrasesInFile" initialized ;
"NumberOfPhrasesInFile" incremented ;
return of the phrase count / assigned to global variable ;

Alternative solutions which include all or some of the following:

- declaring a dynamic array; **A** by implication if supported in language
opening file / specifying the file;
read entire text file into string;
split string into array;
closing file;
read size of array;
return of the phrase count / assigned to global variable;
N.B. More than one mark may be awarded if command combines multiple functions e.g. `ReadAllLines` which opens (1) and closes (1) file, reads entire text file (1) and splits into an array (1) is worth 4 marks

- Solutions which (do not require the loop structure and) **compute the number of phrases from object methods**.
The table below is an indicative (but not exhaustive) list so you need to check any other feasible answers you see, particularly if the screen shot appears to work.

Max 7

Table 1 shows some of the methods for the supported languages which will be used for an alternative solution.

Table 1

List of commands/methods

Language	Function to read entire text file into a string or array*	Function to split string into an array	Function to return array length
Visual Basic 6	ReadAll [1 – read all phrases into string]	`Split` [1]	`UBound` [1]
.NET languages: VB C# Delphi Java	`ReadToEnd` [1 - read all phrases into string] `ReadAllText` [3 - 1 open, 1 close, 1 read all phrases into string] `ReadAllLines` [4 - 1 open, 1 close, 1 read all phrases, 1 split into array]	`Split` [1] except if this mark already given for `ReadAllLines`	`Ubound` [1] `GetUpperBound` [1]
PHP	`File` [4 - 1 open, 1 close, 1 read all phrases, 1 split into array] `File Get Contents` [3 - 1 open, 1 close, 1 read all phrases into string]	`Explode`, `Split` (with some close variations e.g. `Split Split` [1] except if this mark already given for File	`Count` [1]
Java	`Scanner` with delimiter '\\z' [1 – read all phrases into string]	`Split` [1]	`Length` [1]
Python	`Read` [1 – read all phrases into string] `ReadLines` [2 – read all phrases and split into list]	`Split` [1]	`Shape/Len` [1]

** Note that some of the commands in the second column are worth more than one mark as they perform multiple tasks e.g.* File_Get_Contents *in PHP opens and closes the file and reads all the phrases into a string so is worth 3 marks, as shown in []. To answer 8ai the candidate would then need to use* Split/Explode *to break this string up into an array then* Count *to see how many elements there are in the array – i.e. how many phrases were loaded.*

4	4	**** SCREEN CAPTURE ***** *This is conditional on some code for (a) (i)* reports the number of phrases in the file - 24 (**A.** 25) ;	**1**

4	5	**Visual Basic**
		`Sub GenerateRandomPhraseNumber()`
		`' uses global variables NumberOfPhrasesInFile and PhraseNumber`
		`Randomize()`
		`ThisPhraseNumber = Int(Rnd() * NumberOfPhrasesInFile) + 1`
		`End Sub`
		
		Pascal
		`Procedure GenerateRandomPhraseNumber;`
		`{ uses global variables NumberOfPhrasesInFile and PhraseNumber }`
		`Begin`
		`Randomize;`
		`PhraseNumber:=Trunc(Random(NumberOfPhrasesInFile))+1;`
		`End;`
		
		Alternative Implementations
		NB Several alternative implementations possible for both Pascal and Visual Basic
		
		e.g. Pascal
		`Procedure GenerateRandomPhraseNumber (Var NumberOfPhrasesInFile:Integer);`
		`Function GenerateRandomPhraseNumber : Integer;`
		`Function GenerateRandomPhraseNumber (Var NumberOfPhrasesInFile:Integer) : Integer;`
		
		Mark as follows:
		Correct use of the RANDOM/RND function/class with "NoOfPhrasesInFile") ;
		correct range generated (from 1 to "NoOfPhrasesInFile");
		final answer is int**e**ger // implied by variable declaration /return value from a function ;
		
		Note: Commentary in the 'C specific' MS
			3

4	6	***** SCREEN CAPTURE 1 ***** displays phrase number ; ***** SCREEN CAPTURE 2 ***** displays phrase number ;	**2**

4	7	**Visual Basic**
		`Sub SelectPhraseFromFile()`
		`' uses global variable PhraseNumber`
		`Dim Counter As Integer`
		`Dim Found As Boolean`
		`Dim ThisPhraseFromFile As String`
		`Counter = 1`
		`Found = False`
		`FileOpen(1, "MyPhrases.txt", OpenMode.Input)`
		`Do`
		`ThisPhraseFromFile = LineInput(1)`
		`If Counter = PhraseNumber Then`
		`Found = True`
		`Else`
		`Counter = Counter + 1`
		`End If`
		`Loop Until Found = True Or EOF(1)`
		`FileClose(1)`
		`End Sub`
		OR equivalent using the FileStream object and StreamReader method.
		
		Pascal
		`Procedure SelectPhraseFromFile;`
		`{ uses global variable PhraseNumber }`
		`Var`
		`Counter:Integer;`
		`MyPhrasesPipe : TextFile;`
		`ThisPhraseFromFile : String;`
		`Begin`
		`Assign(MyPhrasesPipe, 'MyPhrases.txt');`
		`Reset(MyPhrasesPipe);`
		`Counter:=0;`
		`While (Not Eof(MyPhrasesPipe)) And (Counter<>PhraseNumber)`
		`Do`
		`Begin`
		`Readln(MyPhrasesPipe, ThisPhraseFromFile);`
		`Counter:=Counter+1;`
		`End;`
		`Close(MyPhrasePipe);`
		`End;`
		
		Mark as follows:
		File opened ;
		Loop (post or pre-condition) / FOR-ENDFOR ;
		Counter initialized ;
		Read next phrase from file ;	

		Stored in a temporary variable ; File closed ; Return of the phrase / assigned to global variable ; For loop only … For 1 TO X ; Conditional loop only … Counter incremented ; Boolean variable for trigger / Counter compared with PhraseNumber for trigger ; Boolean variable set to True when located // terminated correctly ; **Alternative solution if entire text file read at once:** - **declaring a dynamic array**; **A** by implication if supported in language opening file / specifying the file; read entire text file into string; split string into array; closing file; access correct cell in array; return of the phrase / assigned to global variable; **N.B.** More than one mark may be awarded if command combines multiple functions e.g. `ReadAllLines` which opens (1) and closes (1) file, reads entire text file (1) and splits into an array (1) is worth 4 marks - **solutions which use object methods** As for Question 8 (a)(ii), look for solutions which compute the phrase in this way. Refer to **Table 1** shown with 8(a)(i).	**Max 7**

4 8 **** SCREEN CAPTURE 1 ****

**** SCREEN CAPTURE 2 ****

Evidence for two different words selected ;

1(0)	**MANCHESTER UNITED**
2(1)	**YELLOW SUBMARINE**
3(2)	**HIP HOP MUSIC**
4(3)	**DETERMINATION**
5 (4)	**PABLO PICASSO**
6 (5)	**THE GRAND CANYON**
7 (6)	**BRICK LANE**
8 (7)	**WIGAN ATHLETIC**
9 (8)	**WORLD MUSIC**
10 (9)	**THE COLISEUM**
11 (10)	**WAR AND PEACE**
12 (11)	**VIVIENNE WESTWOOD**
13 (12)	**EAST ENDERS**
14 (13)	**GRIZZLY BEAR**
15 (14)	**NEW ZEALAND**
16 (15)	**KATE WINSLET**
17 (16)	**THE SUNDAY TIMES**
18 (17)	**THE GUARDIAN**
19 (18)	**HOCKEY STICKS**
20 (19)	**CORONATION STREET**
21 (20)	**GLASTONBURY FESTIVAL**
22 (21)	**SERENDIPITOUS**
23 (22)	**FORTUITOUS**
24 (23)	**FASHION STATEMENT**

2

4 9

Visual Basic

```
Dim NumberOfPhrasesInFile As Integer
Dim PhraseNumber As Integer
Dim ThisPhraseFromFile As String
```

Pascal

```
Var
  NumberOfPhrasesInFile : Integer;
  PhraseNumber : Integer;
  ThisPhraseFromFile : String;
```

	Mark as follows: declare `NumberOfPhrasesInFile` / `PhraseNumber` / `ThisPhraseFromFile` or any plausible variable (MAX 1) ; correct matching plausible data type (MAX 1) ; **Python only:** Data type is implied by assignment e.g. PhraseNumber = 0 **A**. if complete code listing given and additional variable is identified	**MAX 2**

JAVA

1	**8**	`class Question4 {`
		
		`Console console = new Console();`
		`String newWord = "";`
		`String userWordGuess;`
		
		`public Question4(){`
		`newWord=console.readLine("The new word?");`
		`userWordGuess=console.readLine("Your guess?");`
		`if(userWordGuess.equals(newWord)){`
		`console.println("CORRECT");`
		`} else {`
		`console.println("INCORRECT");`
		`} // end if/else`
		`} // end constructor`
		
		`public static void main(String[] args) {`
		`new Question4();`
		`System.exit(0);`
		`} // end Main`
		`} // end Question4`
			Max 7

3	**9**	`private void displayMenu() {`
		
		`console.println("_____");`	

```
        console.println();
        console.println("1. SETTER - Makes new
word/phrase");
        console.println();
        console.println("2. USER - Next letter
guess");
        console.println();
        console.println("3. USER - Make a complete
word/phrase guess");
        console.println();
        console.println("5. End");
        console.println();
    } // end method displayMenu
```

Max 2

4	0

```
    private void inputUsersCompletePhraseGuess() {
        console.println("Procedure
inputUsersCompletePhraseGuess has been called");
    } // end inputUsersCompletePhraseGuess
```

3

4	1

```
     if (choice == 3) {
                inputUsersCompletePhraseGuess();
            } // end if
```

2

4	3

```
  private void countPhrasesFromFile() {
        String fileNameIn = "MyPhrases.txt";
        String newLine;
        numberOfPhrasesInFile = 0;
        try {
            BufferedReader phrasesFile = new
BufferedReader(new FileReader(fileNameIn));

            while ((newLine =
phrasesFile.readLine()) != null) {
                numberOfPhrasesInFile =
numberOfPhrasesInFile + 1;
            } // end while
            phrasesFile.close();
        } catch (IOException e) {
            System.out.println(e.toString());
            System.exit(0);
        } // end try/catch
        console.println("Number of phrases: " +
numberOfPhrasesInFile);

    } // end countPhrasesFromFile
```

Max 7

4	5	`private void generateRandomPhraseNumber()`
		`{`
		`// .nextInt(n) produces nos [0..n[`
		`phraseNumber = generator.nextInt(numberOfPhrasesInFile) + 1;`
		`} // end generateRandomPhraseNumber`
		
		Alternative implementation:
		`private int generateRandomPhraseNumber() {`
		`return generator.nextInt(numberOfPhrasesInFile) + 1;`
		`}`
			3

4	7	`private void selectPhraseFromFile() {`
		`String fileNameIn = "MyPhrases.txt";`
		
		`int counter = 1;`
		`try {`
		`BufferedReader phrasesFile = new BufferedReader(new FileReader(fileNameIn));`
		`while ((counter != phraseNumber)&((thisPhraseFromFile = phrasesFile.readLine()) != null) ) {`
		`counter = counter + 1;`
		`} // end while`
		`console.println("Phrase/phrase selected is: " + thisPhraseFromFile);`
		`phrasesFile.close();`
		`} catch (IOException e) {`
		`System.out.println(e.toString());`
		`System.exit(0);`
		`} // end try/catch`
		`} // end selectPhraseFromFile`
		
		Mark as follows:
		File opened;
		Loop (FOR, post or pre-condition) used to search for the phrase;
		Counter initialised;
		Counter used to control position in the file;
		Counter incremented;
		
		Test for 'EOF';
		Boolean variable for trigger / counter<>`phraseNumber` for trigger;
		Boolean variable set to true when located;
		File closed;
			Max 7

4	**9**	`int numberOfPhrasesInFile;`
		`int phraseNumber;`
		`String thisPhraseFromFile;`
			Max 2

C#

1	**8**	`using System;`
		`using System.Collections.Generic;`
		`using System.Linq;`
		`using System.Text;`
		
		`namespace Question4`
		`{`
		`class Program`
		`{`
		`public static string NewWord;`
		`public static string UserWordGuess;`
		
		`static void Main(string[] args)`
		`{`
		`Console.Write("The new word? ");`
		`NewWord = Console.ReadLine();`
		`Console.Write("Your guess? ");`
		`UserWordGuess = Console.ReadLine();`
		`if (UserWordGuess == NewWord)`
		`Console.WriteLine("CORRECT");`
		`else`
		`Console.WriteLine("INCORRECT");`
		`Console.ReadLine();`
		`}`
		`}`
		`}`
			Max 7

3	**9**	`public static void DisplayMenu()`
		`{`
		
		`Console.WriteLine("_____");`
		`Console.WriteLine("");`
		`Console.WriteLine("1. SETTER - Makes new word/phrase");`
		`Console.WriteLine("");`
		`Console.WriteLine("2. USER - Next letter guess");`
		`Console.WriteLine("");`
		`Console.WriteLine("3. USER – Make a complete word/phrase guess");`
		`Console.WriteLine("");`
		`Console.WriteLine("5. End");`
		`Console.WriteLine("");`	

```
}
```

2

4 0

```
public static void InputUsersCompleteWordGuess()
{
                              Console.WriteLine("Procedure
      InputUsersCompleteWordGuess has been called");
}
```

3

4 1

```
if  (Choice == 3) {
     InputUsersCompleteWordGuess();
}
```

2

4 3

```
public static void CountPhrasesFromFile()
{
     // uses global variable NumberOfPhrasesInFile
     string TempPhrase;

     TextReader MyPhrases = new
StreamReader("MyPhrases.txt");
     NumberOfphrasesInFile = 0;
     TempPhrase = MyPhrases.ReadLine();
     while (TempPhrase != null)
     {
      NumberOfPhrasesInFile++;
      TempPhrase = MyPhrases.ReadLine();
     }
     MyPhrases.Close();
}
```

Alternative to while loop using do . . .while :

```
public static void CountPhrasesFromFile()
{
     // uses global variable NumberOfPhrasesInFile
     string TempPhrase;
     NumberOfPhrasesInFile = 0;
     TextReader MyPhrases = new
StreamReader("MyPhrases.txt");
     do {
          TempPhrase = MyPhrases.ReadLine();
          if (TempPhrase) != null)
               NumberOfPhrasesInFile++;
     } while (TempPhrase != null);

          MyPhrases.Close();
```

		}
		
		Alternative implementation :
		public static int CountPhrasesFromFile()
		{
		. . . . . .
		return NumberOfPhrasesInFile;
		}
		
		A solution which does not require storing in a temporary variable:
		public static void CountPhrasesFromFile()
		{
		// NumberOfPhrasesInFile is a global variable
		TextReader MyPhrases = new StreamReader("MyPhrases.txt");
		NumberOfPhrasesInFile = 0;
		
		while (MyPhrases.ReadLine() != null)
		{
		NumberOfPhrasesInFile++;
		}
		MyPhrases.Close();
		}
			Max 7

4	**5**	public static void GenerateRandomPhraseNumber()
		{
		// uses global variables : NumberOfPhrasesInFile and PhraseNumber
		Random random = new Random();
		PhraseNumber = random.Next(1, NumberOfPhrasesInFile);
		}
		
		Alternative implementations :
		public static void GenerateRandomPhraseNumber(int NumberOfPhrasesInFile)
		
		{
		Random random = new Random();
		PhraseNumber = random.Next(1, NumberOfPhrasesInFile);
		}
		
		public static int GenerateRandomPhraseNumber()
		{	

```
    Random random = new Random();
    PhraseNumber = random.Next(1,
NumberOfPhrasesInFile);
    return PhraseNumber;
}
```

public static int GenerateRandomPhraseNumber

(int NumberOfPhrasesInFile)

```
{
    Random random = new Random();
    PhraseNumber = random.Next(1,
NumberOfPhrasesInFile);
    return PhraseNumber;
}
```

In both of the last two alternatives the *two* statements :

```
    PhraseNumber = random.Next(1,
NumberOfPhrasesInFile);
    return PhraseNumber;
```

can be replaced by the *single* statement:

```
    return random.Next(1,
NumberOfPhrasesInFile);
```

3

4 **7**

```
public static void SelectPhraseFromFile()
{
    // uses global variable PhraseNumber
    int Counter;
    string ThisPhraseFromFile;

    Counter = 1;
    TextReader MyPhrases = new
StreamReader("MyPhrases.txt");
    ThisPhraseFromFile = MyPhrases.ReadLine();
    while (ThisPhraseFromFile != null && Counter
!= PhraseNumber)
    {
         Counter++;
         ThisPhraseFromFile =
MyPhrases.ReadLine();
    }
    MyPhrases.Close();
    Console.WriteLine("Phrase/phrase selected is
{0}",

ThisPhraseFromFile);
}
```

Alternative to while loop using do . . . while :

```
public static void SelectPhraseFromFile()
{
    // uses global variable PhraseNumber
    int Counter;
    string ThisPhraseFromFile;

    Counter = 0;
    TextReader MyPhrases = new
StreamReader("MyPhrases.txt");
    do {
        ThisPhraseFromFile =
MyPhrases.ReadLine();
        Counter++;
    } while (ThisPhraseFromFile != null && Counter
!= PhraseNumber);
    MyPhrases.Close();
    Console.WriteLine("Phrase/phrase selected is
{0}",

ThisPhraseFromFile);
}
```

Alternative implementation using Boolean variable :

```
public static void SelectPhraseFromFile()
{
    // uses global variable PhraseNumber
    int Counter;
    bool Found;

    Counter = 1;
    Found = false;
    TextReader MyPhrases = new
StreamReader("MyPhrases.txt");
    ThisPhraseFromFile = MyPhrases.ReadLine();
    while (!Found && ThisPhraseFromFile != null)
    {
     if (Counter == PhraseNumber)
         Found = true;
     else
     {
         Counter++;
         ThisPhraseFromFile =
MyPhrases.ReadLine();
     }
    }
    MyPhrases.Close();
```

		Console.WriteLine("Phrase/phrase selected is {0}",
		
		ThisPhraseFromFile);
		}
		
		Alternative implementation using Boolean variable and do . . . while :
		public static void SelectPhraseFromFile()
		{
		// uses global variable PhraseNumber
		int Counter;
		bool Found;
		
		Counter = 1;
		Found = false;
		TextReader MyPhrases = new StreamReader("MyPhrases.txt");
		do {
		ThisPhraseFromFile = MyPhrases.ReadLine();
		if (ThisPhraseFromFile != null && Counter == PhraseNumber)
		Found = true;
		else
		Counter++;
		} while (!Found && ThisPhraseFromFile != null)
		MyPhrases.Close();
		Console.WriteLine("Phrase/phrase selected is {0}",
		
		ThisPhraseFromFile);
		}
			Max 7

4	**9**	class Program { int NumberOfPhrasesInFile; int PhraseNumber; string ThisPhraseFromFile;	**Max 2**

***Note*:**
using System.IO
must be added at start of program for text file manipulation in Question 8

C

1	**8**	#include <stdio.h> #include <conio.h> #include <string.h>	

```
void main(void){
   char NewWord[53];
   char UserWordGuessed[53];
   int TheSame;

   printf("The new word?\n");
   flushall();         // to empty keyboard buffer
not essential
   gets(NewWord);
   printf("Your guess\n");
   gets(UserWordGuessed);
   TheSame = strcmp(NewWord,UserWordGuessed);
   if(TheSame = = 0){
       printf("CORRECT\n");
   }
   else {
      printf("INCORRECT\n");
   }
   getch();
}
```

Comparing Strings:

In C you cannot compare 2 strings using if(string1 == string2) Candidates have to use strcmp() which returns 0 (zero) if the 2 strings are the same to a variable on the left of the statement. I have used TheSame to hold the result.

The if(var == 0) could be used instead of the if(str1 == str2) from the Pascal or VB solution answers.

Input/Output:

Some candidates may use #include <iostream.h> and then instead of using printf and scanf functions will use cin >> and cout<< instead which are C++ equivalents see below again using MS Visual C++ Express

```
#include <iostream>
#include <conio.h>
#include <string.h>

using namespace std;

void main(void) {
   char NewWord[53];
   char UserWordGuessed[53];
   int TheSame;
   cout << "The new word?"  << endl;
   cin >> NewWord;
   cout << "Your guess" << endl;
   cin >> UserWordGuessed;
   TheSame = strcmp(NewWord,UserWordGuessed);
   if(TheSame == 0){
       cout << "CORRECT" << endl;
```

		}
		else {
		cout << "INCORRECT" << endl;
		}
		getch();
		}
			Max 7

3	**9**	void DisplayMenu(){
		printf("_____");
		printf("\n");
		printf("1. SETTER - Makes new word/phrase");
		printf("\n");
		printf("2. USER - Next letter guess");
		printf("\n");
		printf("3. USER - Make a complete word/phrase guess");
		printf("\n");
		printf("5. End");
		printf("\n");
		}
		
		OR …Again the candidate may be using cout << instead of printf to output text the two are equivalent and the candidate should use
		cout << "3. USER – Make a complete word/phrase guess" ;
		cout << endl;
			2

4	**0**	void InputUsersCompletePhraseGuess(void){
		printf("Procedure InputUsersCompletePhraseGuess has been called\n");
		}
		
		OR …Again cout << and printf are equivalent i.e.
		
		cout << "Procedure InputUsersCompletePhraseGuess has been called\n"
		or
		cout << "Procedure InputUsersCompletePhraseGuess has been called" << endl;
			3

4	**1**	if(Choice == 3) {
		InputUsersCompletePhraseGuess();
		}
		
		Call to InputUsersCompletePhraseGuess();
		IF statement for Choice 3;
			2

4	**3**	`int CountPhrasesFromFile(void){`
		`  char FileName[15];`
		`  char TempPhrase[53]={'/0'};  // init to zero length`
		`  int NumberOfPhrasesInFile;`
		
		`  strcpy(FileName,"MyPhrases.txt");`
		`  NumberOfPhrasesInFile = 0;`
		`  Fileptr =fopen(FileName,"r");`
		`  fgets(TempPhrase,53,Fileptr);`
		`  NumberOfPhrasesInFile++;`
		`  while(!feof(Fileptr)){`
		`  fgets(TempPhrase,53,Fileptr);`
		`  if(!feof(Fileptr)){`
		`  NumberOfPhrasesInFile++;`
		`  }`
		`  }`
		`  fclose(Fileptr);`
		`  return NumberOfPhrasesInFile;`
		`}`
			Max 7

4	**5**	`int GeneratephraseNumber(void) {`
		`  unsigned int ChosenNumber ;`
		`  unsigned int number;`
		
		`  ChosenNumber = 0;`
		`  number = rand_s(&ChosenNumber);`
		`  while(ChosenNumber >24){`
		`  ChosenNumber -=24;`
		`  }`
		`  return ChosenNumber;`
		`}`
		C Specific
		Randomize function is used or has #define _CRT_RAND_S is used which randomises within the library code; Method used with MS C++ Express edition.
		Method above uses rand_s (to create a randon value in ChosenNumber but this needs to be reduced to a value between 1 and 24 with the while loop. There are many other ways of doing this that are valid.
			3

4	**7**	`char* SelectPhraseFromFile(void){`
		`  char FileName[15];`
		`  char TempPhrase[53];`
		`  int count;`
		
		`  strcpy(FileName,"MyPhrases.txt");`
		`  Fileptr =fopen(FileName,"r");`
		`  count=0;`	

		while((count<ChosenPhraseNumber)&&(!feof(Fileptr))){
		fgets(TempPhrase, 53 ,Fileptr);
		count++;
		}
		fclose(Fileptr);
		return TempPhrase;
		}
			Max 7

4	**9**	Declare NumberOfPhrases/phraseNumber/ThisPhraseFromFile as a global variable(MAX 1); Correct matching of data type (MAX 1); C++ int NumberOfPhrasesInFile; int PhraseNumber; char ThisPhraseFromFile[53]; or other sensible value in square brackets;	**Max 2**

PHP

1	**8**	<?php
		fwrite(STDOUT, "The new word? ");
		$NewWord = trim(fgets(STDIN));
		fwrite(STDOUT, "Your guess? ");
		$UserWordGuess = trim(fgets(STDIN));
		if ($NewWord == $UserWordGuess)
		fwrite(STDOUT, "CORRECT\n");
		else
		fwrite(STDOUT, "INCORRECT\n");
		fgetc(STDIN);
		?>
			Max 7

3	**9**	function DisplayMenu()
		{
		fwrite(STDOUT,
		"_____\n");
		fwrite(STDOUT, "\n");
		fwrite(STDOUT, "1. SETTER - Makes new word/phrase\n");
		fwrite(STDOUT, "\n");
		fwrite(STDOUT, "2. USER - Next letter guess\n");
		fwrite(STDOUT, "\n");
		fwrite(STDOUT, "3. USER – Make a complete word/phrase guess\n");
		fwrite(STDOUT, "\n");	

		fwrite(STDOUT, "5. End\n");
		fwrite(STDOUT, "\n");
		}
			2

4	**0**	function InputUsersCompletePhraseGuess()
		{
		
		fwrite(STDOUT, "Procedure InputUsersCompletePhraseGuess has been called\n");
		}
			3

4	**1**	if ($Choice == 3)
		{
		InputUsersCompletePhraseGuess();
		}
			2

4	**3**	function CountPhrasesFromFile(){
		// NumberOfPhrasesInFile is a global variable
		global $NumberOfPhrasesInFile;
		$NumberOfPhrasesInFile = 0;
		$MyPhrases = fopen("MyPhrases.txt", "r");
		$TempPhrase = fgets($MyPhrases);
		while (!feof($MyPhrases)) {
		$NumberOfPhrasesInFile++;
		$TempPhrase = fgets($MyPhrases);
		}
		fclose($MyPhrases);
		}
		
		Alternative to while loop using do . . . while :
		function CountPhrasesFromFile(){
		// NumberOfPhrasesInFile is a global variable
		global $NumberOfPhrasesInFile;
		$NumberOfPhrasesInFile = 0;
		$MyPhrases = fopen("MyPhrases.txt", "r");
		do {
		$TempPhrase = fgets($MyPhrases);
		if (!feof($MyPhrases))
		$NumberOfPhrasesInFile++;
		} while (!feof($MyPhrases))
		fclose($MyPhrases);
		}
			Max 7

4	**5**		

		function GenerateRandomPhraseNumber(){
		// uses global variables NumberOfPhrasesInFile and PhraseNumber
		global $NumberOfPhrasesInFile;
		global $PhraseNumber;
		$PhraseNumber = rand(1, $NumberOfPhrasesInFile);
		}
		
		Alternative implementations:
		function GenerateRandomPhraseNumber($NumberOfPhrasesInFile)
		{
		global $PhraseNumber;
		$PhraseNumber = rand(1, $NumberOfPhrasesInFile);
		}
		
		function GenerateRandomPhraseNumber(){
		global $NumberOfPhrasesInFile;
		$PhraseNumber = rand(1, $NumberOfPhrasesInFile);
		return $PhraseNumber
		}
		
		function GenerateRandomPhraseNumber($NumberOfPhrasesInFile)
		{
		$PhraseNumber = rand(1, $NumberOfPhrasesInFile);
		return $PhraseNumber;
		}
		
		In both of the last two alternatives, the *two* statements :
		$PhraseNumber = rand(1, $NumberOfPhrasesInFile);
		return $PhraseNumber;
		can be replaced by the *single* statement :
		return rand(1, $NumberOfPhrasesInFile);
			3

4	**7**	function SelectPhraseFromFile(){
		global $PhraseNumber;
		$Counter = 1;
		$MyPhrases = fopen("MyPhrases.txt", "r");
		$ThisPhraseFromFile = fgets($MyPhrases);
		while (!feof($MyPhrases) && Counter != PhraseNumber)
		{	

		$Counter++;
		$ThisPhraseFromFile = fgets($MyPhrases);
		}
		fclose($MyPhrases);
		fwrite(STDOUT, "Phrase/phrase selected is " .
		
		$ThisPhraseFromFile);
		}
		
		Alternative using do . . . while :
		
		function SelectPhraseFromFile(){
		global $PhraseNumber;
		$Counter = 0;
		$MyPhrases = fopen("MyPhrases.txt", "r");
		do
		{
		$ThisPhraseFromFile =
		fgets($MyPhrases);
		$Counter++;
		} while (!feof($MyPhrases) && Counter !=
		PhraseNumber)
		fclose($MyPhrases);
		fwrite(STDOUT, "Phrase/phrase selected is " .
		
		$ThisPhraseFromFile);
		}
		
		Alternative using Boolean variable :
		function SelectPhraseFromFile(){
		global $PhraseNumber;
		$Counter = 1;
		$Found = false;
		$MyPhrases = fopen("MyPhrases.txt", "r");
		while(!$Found && !feof($MyPhrases)){
		$ThisPhraseFromFile = fgets($MyPhrases);
		if ($Counter == $PhraseNumber) {
		$Found = true;
		}
		else
		$Counter++;
		}
		fclose($MyPhrases);
		fwrite(STDOUT, "The phrase/phrase selected is
		..".$ThisPhraseFromFile);
		}
			Max 7

4	**9**	global $NumberOfPhrasesInFile; global $PhraseNumber;	

		global $ThisPhraseFromFile; - declared at start of *main* program	**Max 2**

PYTHON

1	**8**	NewWord = raw_input("The new word?")
		UserWordGuess = raw_input("Your Guess?")
		if UserWordGuess == NewWord:
		    print "CORRECT"
		else:
		    print "INCORRECT"
		raw_input() # keep window on screen
			Max 7

3	**9**	def DisplayMenu():
		    print "_____"
		    print ""
		    print "1. SETTER - Makes new word/phrase"
		    print ""
		    print "2. USER - Next letter guess"
		    print ""
		    print "3. USER - Make a complete word/phrase guess"
		    print ""
		    print "5. End"
		    print ""
			2

4	**0**	def InputUsersCompletePhraseGuess():
		    print "Procedure InputUsersCompletePhraseGuess has been called"
		    raw_input()
		
		allow missing raw_input() – only keeps window open.
		
		(NB no explicit "end" statement as in Pascal / VB) –
		Award mark for correct indentation of print statement.
			3

4	**1**	elif Response == '3': InputUsersCompletePhraseGuess() Inverted commas needed to indicate string value as returned by raw_input() function	**2**

4	**3**	def CountPhrasesFromFile1():
		    global NumberOfPhrasesInFile
		    f = open('MyPhrases.txt','r')
		    AllPhrases = f.readlines()	

		NumberOfPhrasesInFile = len(AllPhrases)
		f.close()
		
		or
		
		def CountPhrasesFromFile2():
		global NumberOfPhrasesInFile
		f = open('MyPhrases.txt','r')
		NumberOfPhrasesInFile = 0
		for phrase in f.readlines():
		NumberOfPhrasesInFile = NumberOfPhrasesInFile + 1
		f.close()
		
		Accept NumberOfPhrasesInFile += 1
			Max 7

4	**5**	Needs "import random" declared at start of program def GenerateRandomPhraseNumber(): global PhraseNumber, NumberOfPhrasesInFile PhraseNumber = random.randrange(NumberOfPhrasesInFile)	**3**
4	**7**	def SelectPhraseFromFile(): global PhraseNumber, ThisPhraseFromFile f = open('MyPhrases.txt','r') Phrases = f.readlines() ThisPhraseFromFile = Phrases[PhraseNumber] print "The Phrase selected is ... %s" % ThisPhraseFromFile or print "The Phrase selected is ... ", ThisPhraseFromFile f.close()	**Max 7**

4	**9**	Declare NumberOfPhrasesInFile / PhraseNumber and initialise at start of program to assign data type. NumberOfPhrasesInFile = 0 PhraseNumber = 0 ThisPhraseFromFile = ‘’	**Max 2**