GCSE
COMPUTER SCIENCE

Paper 1 Supplementary questions

These supplementary questions are taken from the 2014/5/6 GCSE Computer Science (4512) assessments. The table on page 2 shows the content in our new GCSE Computer Science (8520) specification to which these questions relate. These supplementary questions should not be treated as a complete paper, they do not provide a balanced coverage of the specification or the assessment objectives in the same way that a fully live paper would do.

It is hoped that teachers will find these questions to be a useful resource to enable them to understand the nature of questions that could be assessed as part of the specification.

Version 1.1
28/02/17

8520 Specification Reference	Question from 4512 – June 2014	Question from 4512 – June 2015	Question from 4512 – June 2016
3.1 Fundamentals of algorithms 3.1.1 Representing algorithms (Pages 3 -25)	12, 7(a), 7(b)	3(a), 3(b)(i), 3(b)(ii), 3(b)(iii), 7(c), 9, 10(a), 10(b)	3(b), 6(d)
3.1.2 Efficiency of algorithms (Pages 26-27)		3(b)(iv)	
3.2 Programming 3.2.1 Data types (Pages 28-31)		7(b)	2(b), 2(e), 6(a)
3.2.2 Programming concepts (Pages 32-41)	3(b), 3(c), 3(d)		6(e), 10(a), (b), (c), (d) and (e)
3.2.6 Data structures (Pages 42-43)		2(a), 2(b)	
3.2.10 Subroutines (procedures and functions) (Pages 44-48)	3(a), 3(e)	7(a), 7(d)	6(d), (e), (f)
3.2.12 Robust and Secure programming (Page 49)			3(a) (i) and (ii)
3.3 Fundamentals of data representation 3.3.1 Number bases (Page 50)	1(c)		
3.3.2 Converting between number bases (Pages 51-55)	1(a), 1(b), 1(d)	1(a), 1(b), 1(c)	1(a), 1(b), 1(c) 1(d)
3.3.3 Units of information (Page 56)		1(d)	
3.3.5 Character encoding (Page 57)		1(e)	2(d)
3.3.6 Representing images (Pages 58-60)	1(f)	1(f)	
3.3.7 Representing sound (Page 61)	1(e)		
3.3.8 Data compression (Page 62)	1(d)		
3.4 Computer systems 3.4.4 Systems architecture (Pages 63-75)	2(a), 2(b), 2(c), 5(b), 6	6(a), 6(b), 6(c), 6(d)	1(f)(i), (f)(ii), 8

Answer **all** questions in the spaces provided.

Topic: 3.1 Fundamental of algorithms
3.1.1 Representing algorithms

Questions and Mark Scheme from 4512 – June 2014

12 The following algorithm determines the number of carriages a train will need.

The array called `passengers` is used to store the change in the number of passengers at each train station. For example, at the first stop the total number of passengers increases by 100, at the second stop the total number of passengers decreases by 20, at the third stop the total number of passengers increases by 70 and so on.

Note: array indexing starts at 1.

```
passengers ← [100, -20, 70, -50, -100]
carriages ← 0
total ← 0
max ← 0
index ← 1
WHILE index ≤ 5
  total ← total + passengers[index]
  IF total > max THEN
    max ← total
  ENDIF
  index ← index + 1
ENDWHILE
carriages ← max / 50
```

12 (a) Complete the trace table (**Table 6**) for this program.

[6 marks]

Table 6

carriages	total	max	index
0	0	0	1

12 (b) Write an algorithm (using either **pseudocode** or a **flowchart**) that calculates the amount of fuel a train will need to complete a journey. The algorithm must:

- ask the user how many kilometres the journey will be
- only continue if the user enters a value greater than zero
- set the amount of fuel to a number 100 times greater than the number of kilometres
- not allow the amount of fuel to be less than 1500
- finally, display the amount of fuel needed.

[7 marks]

<table>
<tr>
<td>12</td>
<td>a</td>
<td></td>
<td>The correct, completed trace table should look like this:

<table>
<tr><th>carriages</th><th>total</th><th>max</th><th>index</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td></td><td>100</td><td>100</td><td>2</td></tr>
<tr><td></td><td>80</td><td></td><td>3</td></tr>
<tr><td></td><td>150</td><td>150</td><td>4</td></tr>
<tr><td></td><td>100</td><td></td><td>5</td></tr>
<tr><td></td><td>0</td><td></td><td>6</td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
</table>

Marks awarded as follows (do not penalise if values appear on different lines to the above trace table as long as the sequence of values within the column is correct):

1 mark for the index incremented by 1 at each step up to at least 5;

1 mark for the index ending at 6;

1 mark for max set to 100 ;

1 mark for final max value set to 150;

1 mark for all total values correct;

1 mark for carriages changed once to 3//1 mark follow through for carriages changed once to a non-zero number which is the last value of max divided by 50;

I all other values.</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>b</td>
<td></td>
<td>Marks awarded as follows (allow any logically equivalent and correct answer). The marks are labelled A – G and shown in the examples where they are awarded:

1 mark for assigning user input to a variable (permit any variable name); [mark A]

1 mark for using selection to check that a value greater than zero has been entered (two logically equivalent examples are given below although there are many logically equivalent ways to accomplish this); [mark B]

1 mark for the correct expression that multiplies whatever variable is holding the kilometres by 100; [mark C]

1 mark for assigning the value of the above expression to a variable (permit any variable name, the expression need not be correct); [mark D]

1 mark for using selection to check if the amount of fuel used is less than 1500 (two logically equivalent examples are given below although there are many logically equivalent</td>
<td>7</td>
</tr>
</table>

ways to accomplish this); [mark E]

1 mark for assigning the value 1500 to the above variable, or displaying the value 1500, within the selection above (the selection need not be correct); [mark F]

1 mark for outputting the value of the above variable at the end of the algorithm; [mark G]

If the sequence of these marks is incorrect then reward only the higher scoring statement. For example

```
fuel ← km * 100 [D] [C]
km ← USERINPUT [A]
```

The two statements are in the wrong sequence so reward the higher scoring statement (1st line).

Example 1 (italicised square brackets indicate where marks are awarded):

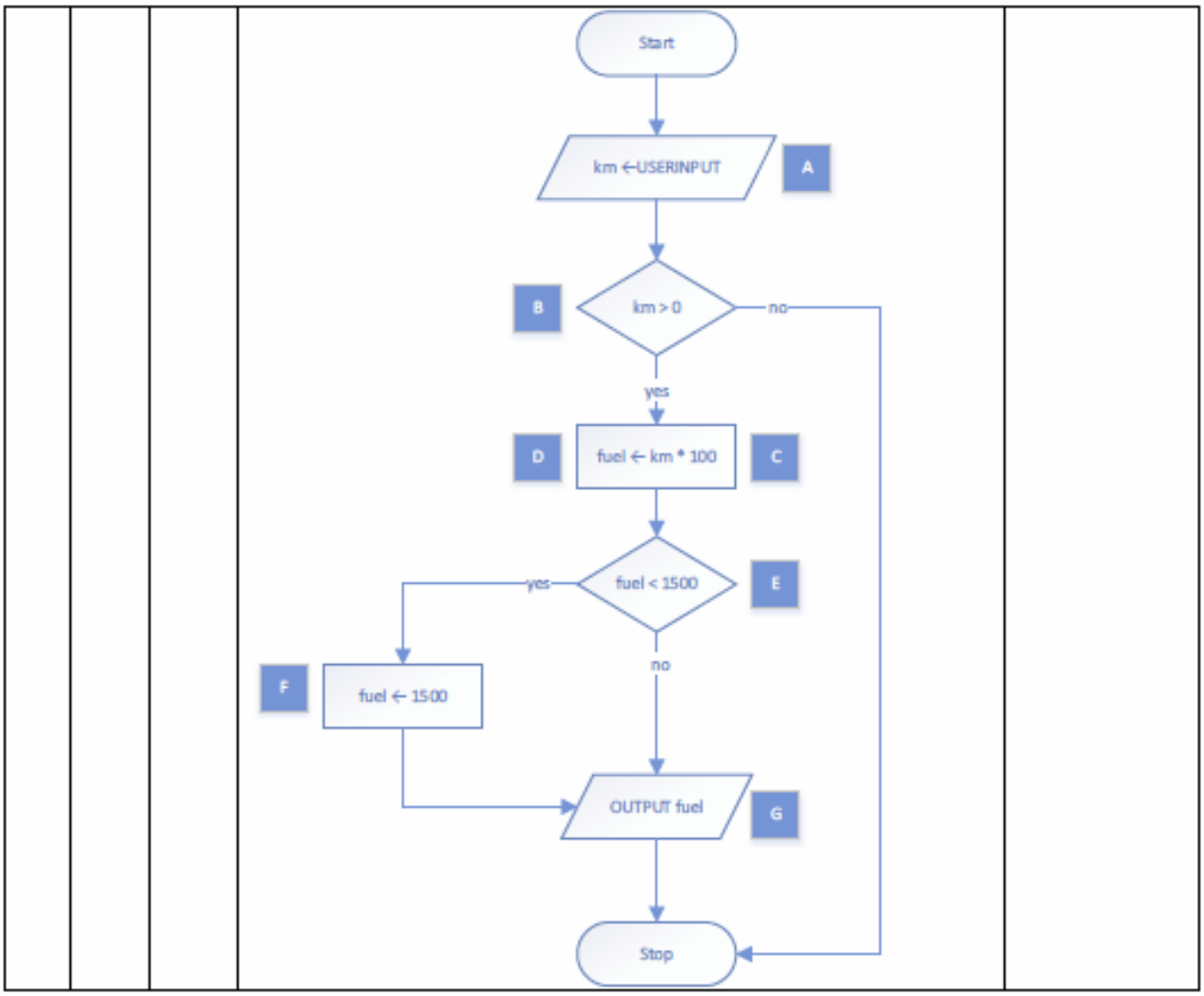
```
km ← USERINPUT [A]
IF km > 0 THEN [B]
  fuel ← [D] 100 * km [C]
  IF fuel < 1500 THEN [E]
    fuel ← 1500 [F]
  ENDIF
  OUTPUT fuel [G]
ENDIF
```

Example 2 (brackets indicate where marks are awarded):

```
km ← USERINPUT [A]
IF km ≤ 0 THEN [B]
  STOP
ELSE
  fuel ← [D] km * 100 [C]
  IF NOT (fuel ≥ 1500) THEN [E]
    fuel ← 1500 [F]
  ENDIF
ENDIF
OUTPUT fuel [G]
```

Example 3 (dark squares indicate where marks are awarded, permit incorrect flowchart shapes although decision boxes must have two labelled arrows coming out for the relevant marks – B and E – to be awarded):

12 (b) continues on the next page.


Start
km ←USERINPUT
A
B
km > 0
no
yes
D
fuel ← km * 100
C
fuel < 1500
E
yes
no
F
fuel ← 1500
OUTPUT fuel
G
Stop

7 Figure 3 shows a function:

Figure 3

```
FUNCTION Compare(x,y)
  IF x > y THEN
    RETURN 1
  ELSE
    IF x < y THEN
      RETURN -1
    ELSE
      RETURN 0
    ENDIF
  ENDIF
ENDFUNCTION
```

7 (a) The function `Compare` returns an integer value.

Explain why a Boolean return value could not have been used.

[1 mark]

.....

.....

.....

7 (b) Each of the following expressions evaluates to an integer. Give the integer value for each:

7 (b) (i) `Compare(4,4)`

[1 mark]

.....

.....

7 (b) (ii) `Compare(1,-1)`

[1 mark]

.....

.....

7	a		Because Boolean only allows two possible return values (and this function requires three).	1
7	b	i	0	1
7	b	ii	1	1
7	b	iii	-1// Allow follow through **only** for the following cases: Answer is 1 if the answer to 7(b)(i) is greater than the answer to 7(b)(ii), or Answer is 0 if the answer to 7(b)(i) is the same as the answer to 7(b)(ii).	1

Question and Mark Scheme from 4512 – June 2015

3 (a) Define the term **algorithm**.

[2 marks]

.....

.....

.....

.....

Question 3 continues on the next page

3 (b) Two algorithms, **Algorithm 1** and **Algorithm 2**, are shown below. Both algorithms have the same purpose.

Note: array indexing starts at 1.

Algorithm 1

```
a ← "diffie"
matched ← false
i ← 0
WHILE i < 5
  i ← i + 1
  IF arr[i] = a THEN
    matched ← true
  ENDIF
ENDWHILE
```

Algorithm 2

```
a ← "diffie"
matched ← false
i ← 0
WHILE i < 5 AND matched = false
  i ← i + 1
  IF arr[i] = a THEN
    matched ← true
  ENDIF
ENDWHILE
```

The completed trace tables for **Algorithm 1** and **Algorithm 2** are shown below when the array `arr` is `["kleene", "diffie", "naur", "karp", "hopper"]`.

matched	i
false	0
	1
true	2
	3
	4
	5

Completed trace table for **Algorithm 1**

matched	i
false	0
	1
true	2

Completed trace table for **Algorithm 2**

3 (b) (i) Both algorithms use a variable called `i` for the same purpose. State the purpose of the variable `i`.

[1 mark]

.....

.....

3 (b) (ii) What is the data type of the variable `matched`?

[1 mark]

.....

.....

3 (b) (iii) **Algorithm 1** and **Algorithm 2** both have the same purpose. State this purpose.

[1 mark]

.....

.....

3	a		A series of instructions//sequence of steps; (Designed to) perform a particular task//solve a problem; **A.** Other wording	**2**
3	b	i	It is an index//counter/stepper (for the array); **A.** Answer that refers to its role in array indexing such as "Used to show which item in the array is the current one."	**1**
3	b	ii	Boolean **A.** Bool (or similar abbreviation) **R.** True/False or Yes/No	**1**
3	b	iii	(The purpose of the algorithms is to) check if an array contains a specific value/the value "`diffie`"/the value of `a`;	**1**

7 The following function calculates the second hand price of different models of car. The parameter `condition` is an integer with a value between 1 and 4 where 1 is excellent and 4 is very bad.

```
FUNCTION CarPrice(model, condition, age)
  cost ← 0

  IF model = 'Daley' THEN
    cost ← 6000
  ELSE
    IF model = 'Minty' THEN
      cost ← 4000
    ELSE
      cost ← 2000
    ENDIF
  ENDIF

  CASE condition OF
    1: cost ← cost - 100
    2: cost ← cost - 300
    3: cost ← cost - 500
    4: cost ← cost - 1000
  ENDCASE

  cost ← cost / age

  RETURN cost
ENDFUNCTION
```

7 (c) Complete the trace table below showing the changes in the variable `cost` when the function `CarPrice` is called with the following arguments:

```
CarPrice('Tidy', 4, 2)
```

cost

[4 marks]

7	**c**		1 mark for every correct row that appears in the correct sequence: **cost** 0 2000 1000 500 Example answers and their associated marks are: **cost** 500 1 mark **cost** 0 500 2 marks	**4**

9 An English teacher wants to estimate how long it should take his students to read a book. You have been asked to develop an algorithm to calculate this estimate. The algorithm must do the following:

- ask the teacher how many pages the book has and store this in an appropriately named variable
- for every page in the book the algorithm should:
  - ask the teacher if the page looks 'easy' or 'difficult'
  - if a page is 'difficult' then the total number of seconds should increase by 100
  - if a page is 'easy' then the total number of seconds should increase by 40
- after the teacher has entered the difficulty level for all the pages, the algorithm should output the estimated number of seconds that it should take to read the book.

Write pseudocode or draw a flowchart that represents this algorithm.

[9 marks]

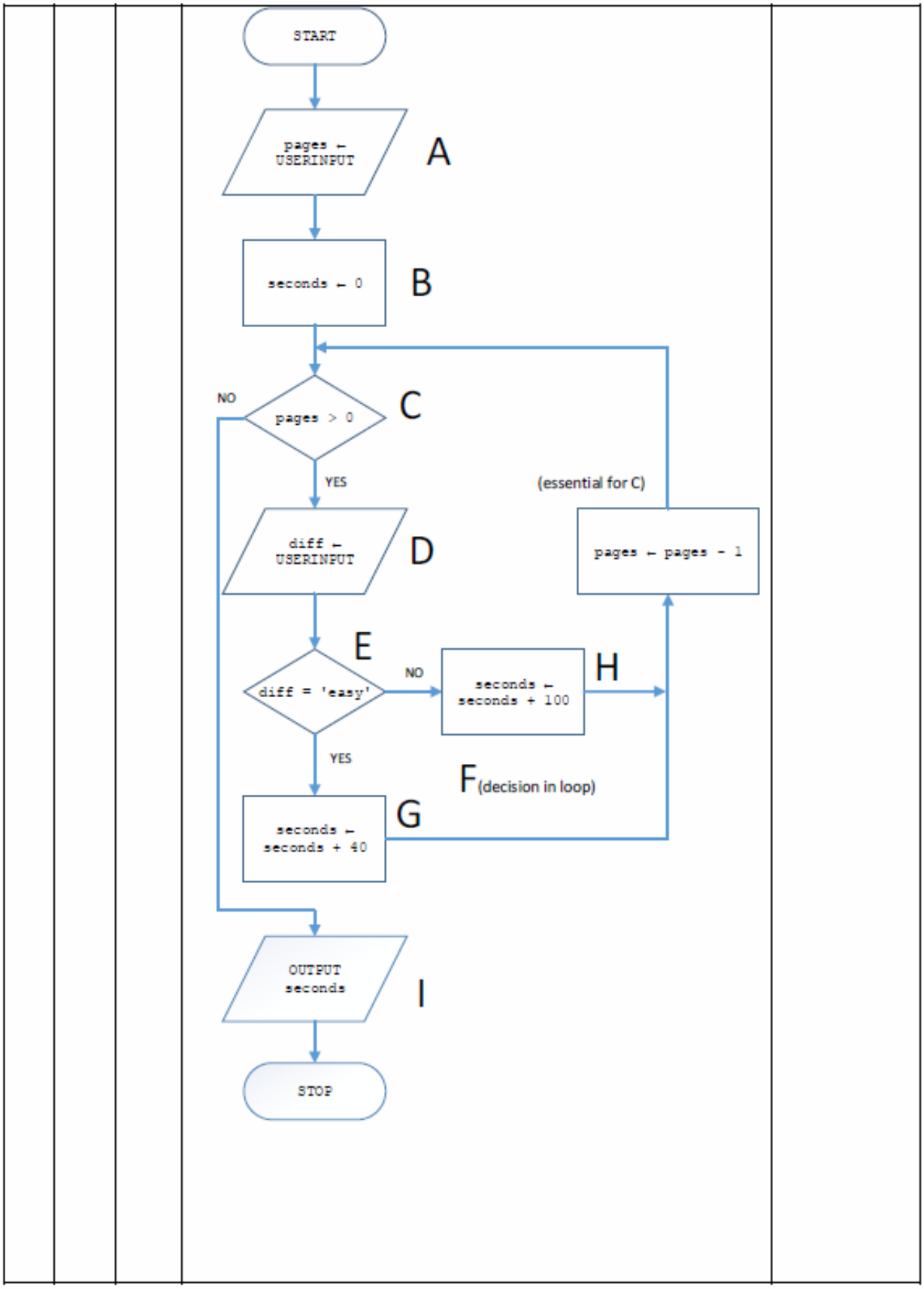
9			Marks awarded as follows (allow any logically equivalent and correct answer). The marks are labelled A – I and shown in the examples where they are awarded:
			A. **1 mark** for assigning user input to a variable (permit any variable name, `pages` has been used in the examples);
			B. **1 mark** for creating a variable that stores the total number of seconds (permit any variable name, `seconds` has been used in the examples) and instantiating this to zero (mark can only be awarded if this is declared **outside** of the loop);
			C. **1 mark** for using a loop to iterate over every page (two logically equivalent examples are given below although there are many logically equivalent ways to accomplish this);
			D. **1 mark** for asking for the user input for the page difficulty; ***(Note that no marks are awarded for validating the user input)***
			E. **1 mark** for using selection to decide if user input is 'easy' (this does not need to be explicit and could possibly the ELSE clause where the IF is asking if it is 'difficult');
			F. **1 mark** for using selection within the loop;
			G. **1 mark** for increasing the number of seconds by 40 within the correct selection block;
			H. **1 mark** for increasing the number of seconds by 100 within the correct selection block;
			I. **1 mark** for outputting the total number of seconds taken **outside** of the loop;
			
			Example 1 (every italicised square bracket indicates where that mark is awarded):
			`pages ← USERINPUT` *[A]*
			`seconds ← 0` *[B]*
			`REPEAT pages` *[C]*
			`  diff ← USERINPUT` *[D]*
			`  IF diff = 'easy' THEN` *[E][F as used within the loop]*
			`    seconds ← seconds + 40` *[G]*
			`  ELSE`
			`    seconds ← seconds + 100` *[H]*
			`  ENDIF`
			`ENDREPEAT`
			`OUTPUT seconds` *[I]*
			Example 2 (every italicised square bracket indicates where that mark is awarded):
			`pages ← USERINPUT` *[A]*
			`seconds ← 0` *[B]*
			`WHILE pages > 0` *[C]*
			`  diff ← USERINPUT` *[D]*
			`  IF diff = 'easy' THEN` *[E][F as used within the loop]*
			`    seconds ← seconds + 40` *[G]*
			`  ENDIF`
				9

```
  IF diff = 'difficult' THEN
    seconds ← seconds + 100 [H]
  ENDIF
  pages ← pages - 1 [essential for mark C]
ENDWHILE
OUTPUT seconds [I]
```

Example 3 (every italicised square bracket indicates where that mark is awarded):

```
pages ← USERINPUT [A]
seconds ← 0 [B]
FOR i ← 1 TO pages [C]
  IF USERINPUT = 'easy' THEN [D][E][F as used within the loop]
    seconds ← seconds + 40 [G]
  ELSE
    seconds ← seconds + 100 [H]
  ENDIF
ENDFOR
OUTPUT seconds [I]
```

Example 4 using a flowchart (large annotated letters indicate where that mark is awarded):


START
pages ← USERINPUT
A
seconds ← 0
B
NO
pages > 0
C
YES
(essential for C)
diff ← USERINPUT
D
pages ← pages - 1
E
diff = 'easy'
NO
seconds ← seconds + 100
H
YES
F (decision in loop)
seconds ← seconds + 40
G
OUTPUT seconds
I
STOP

10 A built-in function commonly found in programming languages is one that finds the character in a string at a specific position. In some programming languages this function is called `CharAt`.

`CharAt(str, i)` returns the character found at position `i` of the string `str`. For example,

```
CharAt("abc", 1)  returns 'a'
CharAt("abc", 3)  returns 'c'
```

10 (a) (i) What value will be returned by the function call `CharAt("hello", 5)`?

[1 mark]

.....

10 (a) (ii) What value will be returned by the function call `CharAt("goodbye", (1+3))`?

[1 mark]

.....

10 (b) A palindrome is a string that is the same read forwards or backwards. For example, "abba" and "abcba" are both palindromes but "abcbb" is not.

The following algorithm uses the function CharAt to check if a string is a palindrome. This algorithm also uses the LEN function. LEN returns the length of a string, for example LEN("cpu") returns 3.

Note: line numbers have been shown but are not part of the algorithm.

```
1   strIn ← USERINPUT
2   isPalindrome ← true
3   iUp ← 1
4   iDown ← LEN(strIn)
5   WHILE iUp < iDown
6     IF CharAt(strIn, iUp) ≠ CharAt(strIn, iDown) THEN
7       isPalindrome ← false
8     ENDIF
9     iUp ← iUp + 1
10    iDown ← iDown - 1
11  ENDWHILE
```

Complete the trace table for this algorithm when the user input is "abcaba".

strIn	isPalindrome	iUp	iDown
abcaba			

[6 marks]

<table>
<tr><td>10</td><td>a</td><td>i</td><td>'o' A. without quote marks</td><td>1</td></tr>
<tr><td>10</td><td>a</td><td>ii</td><td>'d' A. without quote marks</td><td>1</td></tr>
<tr><td>10</td><td>b</td><td></td><td>The complete and correct trace table is:

<table>
<tr><th>strIn</th><th>isPalindrome</th><th>iUp</th><th>iDown</th></tr>
<tr><td>abcaba</td><td>true</td><td>1</td><td>6</td></tr>
<tr><td></td><td></td><td>2</td><td>5</td></tr>
<tr><td></td><td></td><td>3</td><td>4</td></tr>
<tr><td></td><td>false</td><td>4</td><td>3</td></tr>
</table>

1 mark for isPalindrome first value true;
1 mark for isPalindrome last value false;
1 mark for iUp starting at 1;
1 mark for iUp incrementing by 1 and ending at 4;
1 mark for iDown starting at 6;
1 mark for iDown decrementing by 1 and ending at 3;

I. Anything written in the strIn column.</td><td>6</td></tr>
</table>

Question and Mark Scheme from 4512 – June 2016

3 (b) Develop an algorithm using pseudocode or a flowchart that asks the user to create a new password.

The algorithm should:

- get the user to enter a password
- get the user to re-enter the password
- repeat the two bullet points above until both entered passwords are identical
- output "`password created`" when they are identical.

[5 marks]

3	**b**		Marks awarded as follows (allow any logically equivalent and correct answer). The marks are labelled **A – E** and shown in the examples where they are awarded: **A.** 1 mark for assigning user input to a variable (permit any variable name); **B.** 1 mark for assigning the second user input to a distinct variable from that used in **A**, or the second user input used in such a way that doesn't require a variable, e.g. USERINPUT = password1; **C.** 1 mark for using a condition-controlled loop such as a `WHILE` loop with a correct Boolean expression to control this loop (this will depend on the type of loop used but will test for equality of the two passwords); **D.** 1 mark for the user inputting the two passwords again within the loop and assigning these to their respective variables; **E.** 1 mark for outputting `"password created"` at a point in the code where no further user input will occur (**A.** spelling mistakes for `"password created"`);	**5**

Example 1 (italicised square brackets indicate where marks are awarded):

```
password1 ← USERINPUT [A]
password2 ← USERINPUT [B]
WHILE password1 ≠ password2 [C]
  password1 ← USERINPUT
  password2 ← USERINPUT [D with line above]
ENDWHILE
OUTPUT 'password created' [E]
```

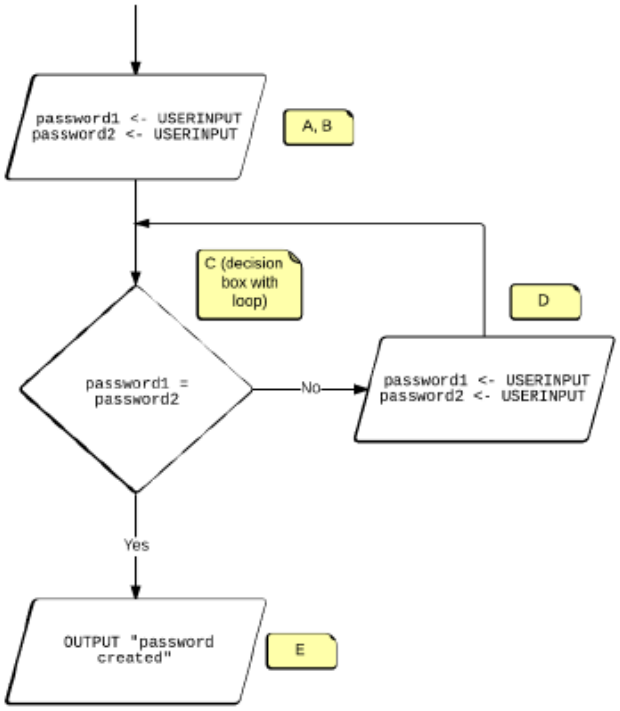
Example 2 (italicised square brackets indicate where marks are awarded):

```
password1 ← USERINPUT [A]
password2 ← USERINPUT [B]
match ← false
IF password1 = password2 THEN
  match ← true
ENDIF
WHILE match = false [C with three lines above and the IF statement within the loop]
  password1 ← USERINPUT
  password2 ← USERINPUT [D with line above]
  IF password1 = password2 THEN
    match ← true
  ENDIF
ENDWHILE
OUTPUT 'password created' [E]
```

Example 3 (italicised square brackets indicate where marks are awarded):

```
match ← false
REPEAT
  password1 ← USERINPUT [A]
  password2 ← USERINPUT [B][D with line above]
  IF password1 = password2 THEN
    match ← true
    OUTPUT 'password created' [E]
  ENDIF
UNTIL match = true [C for condition for DO-WHILE]
```

Example 4 (notes indicate where marks are awarded):


password1 <- USERINPUT
password2 <- USERINPUT
A, B
C (decision box with loop)
D
password1 = password2
No
password1 <- USERINPUT
password2 <- USERINPUT
Yes
OUTPUT "password created"
E

Question and Mark Scheme from 4512 – June 2016

6 (d) Complete the trace table below showing the changes in the variable `x` and the output for the procedure call `Mult(2,3)`.

x	Output

[4 marks]

6	**d**		1 mark for the first value of 1; 1 mark for incrementing each row by 1; 1 mark for the last value of 4; 1 mark for outputs being twice the value of x if at least two values of x are given (**I.** if x is incorrect) The completed correct trace table is: x: 1, 2, 3, 4 Output: 2, 4, 6,	**4**

Topic: 3.1.2 Efficiency of algorithms

Question and Mark Scheme from 4512 – June 2015

3 (b) Two algorithms, **Algorithm 1** and **Algorithm 2**, are shown below. Both algorithms have the same purpose.

Note: array indexing starts at 1.

Algorithm 1

```
a ← "diffie"
matched ← false
i ← 0
WHILE i < 5
  i ← i + 1
  IF arr[i] = a THEN
    matched ← true
  ENDIF
ENDWHILE
```

Algorithm 2

```
a ← "diffie"
matched ← false
i ← 0
WHILE i < 5 AND matched = false
  i ← i + 1
  IF arr[i] = a THEN
    matched ← true
  ENDIF
ENDWHILE
```

The completed trace tables for **Algorithm 1** and **Algorithm 2** are shown below when the array `arr` is `["kleene", "diffie", "naur", "karp", "hopper"]`.

matched	i
false	0
	1
true	2
	3
	4
	5

Completed trace table for **Algorithm 1**

matched	i
false	0
	1
true	2

Completed trace table for **Algorithm 2**

3 (b) (iv) Give **one** reason why **Algorithm 2** could be considered to be a better algorithm.

[1 mark]

.....

.....

3	**b**	**iv**	(Algorithm 2 is a better algorithm because) as soon as a match is made it stops (the while loop)//less matches need to be made//it is more efficient//it stops at the correct index//the value of `i` will be set to the index of the value `a` (`diffie`)	**1**

Topic: 3.2 Programming
3.2.1 Data types

Question and Mark Scheme from 4512 – June 2015

7 The following function calculates the second hand price of different models of car. The parameter `condition` is an integer with a value between 1 and 4 where 1 is excellent and 4 is very bad.

```
FUNCTION CarPrice(model, condition, age)
  cost ← 0

  IF model = 'Daley' THEN
    cost ← 6000
  ELSE
    IF model = 'Minty' THEN
      cost ← 4000
    ELSE
      cost ← 2000
    ENDIF
  ENDIF

  CASE condition OF
    1: cost ← cost - 100
    2: cost ← cost - 300
    3: cost ← cost - 500
    4: cost ← cost - 1000
  ENDCASE

  cost ← cost / age

  RETURN cost
ENDFUNCTION
```

7 (b) Tick the most appropriate data type of the variable `cost`.

Data Type	Tick one box
Boolean	
Character	
Real	
String	

[1 mark]

7	**b**		Real; **R.** If more than one box ticked	**1**

Question and Mark Scheme from 4512 – June 2016

2 (b) Programming languages typically use data types. Explain how one bit could be used to store a Boolean value.

[1 mark]

2	**b**		1 and 0 could represent true and false // A bit and a Boolean data type both have only two values; **A.** other wording that has equivalent meaning.	**1**

2 (e) The following are data types (labelled **A – E**).

A. Integer
B. Boolean
C. Real
D. Character
E. String

For each of the values in the table, write the label of the **most** suitable data type. Use a label only once.

Value	**Label (A – E)**
43.13	
"Curry-Howard"	
978	

[3 marks]

<table>
<tr>
<td>2</td>
<td>e</td>
<td></td>
<td>1 mark for each correct label;

The correct table is:

<table>
<tr><th>Value</th><th>Label (A-E)</th></tr>
<tr><td>43.13</td><td>C</td></tr>
<tr><td>“Curry-Howard”</td><td>E</td></tr>
<tr><td>978</td><td>A</td></tr>
</table>

A. Real instead of C, string instead of E and integer/Int instead of A

R. If a mark is duplicated

For example, this answer would score 1 mark:

<table>
<tr><th>Value</th><th>Label (A-E)</th></tr>
<tr><td>43.13</td><td>C</td></tr>
<tr><td>“Curry-Howard”</td><td>A</td></tr>
<tr><td>978</td><td>A</td></tr>
</table>
</td>
<td>3</td>
</tr>
</table>

Question and Mark Scheme from 4512 – June 2016

6 The pseudocode in **Figure 3** represents a procedure called `Mult`.

Figure 3

```
PROCEDURE Mult(n, m)
  x ← 1
  WHILE x ≤ m
    OUTPUT n * x
    x ← x + 1
  ENDWHILE
ENDPROCEDURE
```

The pseudocode in **Figure 4** represents a procedure called `Display`.

Figure 4

```
PROCEDURE Display(a, b)
  IF b > 3 THEN
    Mult(a, 3)
  ELSE
    Mult(a, b)
  ENDIF
ENDPROCEDURE
```

6 (a) Select the **most** suitable data type for the parameter `n` in the procedure `Mult` (tick **one** box only).

Most suitable data type of n	Tick **one** box
String	
Boolean	
Integer	

[1 mark]

6	**a**		**Most suitable data type of n** / **Tick one box**: String — ; Boolean — ; Integer — ✓	**1**

Topic: 3.2.2 Programming concepts

Question and Mark Scheme from 4512 – June 2014

3 **Figure 1** shows a pseudocode representation of the function called `FindHighest`. `FindHighest` is used to find the largest value stored in an array.

Note: line numbers have been shown but are not part of the function.

Figure 1

```
1        FUNCTION FindHighest(arr)
2          highest ← arr[1]
3          FOR i ← 2 TO LEN(arr)
4            IF arr[i] > highest THEN
5              highest ← arr[i]
6            ENDIF
7          ENDFOR
8          RETURN highest
9        ENDFUNCTION
```

3 (b) This function uses iteration. Give the line number on which iteration **starts**.

[1 mark]

.....

3 (c) This function uses selection. Give the line number on which selection **starts**.

[1 mark]

.....

3 (d) This function uses variable assignment. Give the line number in the function where variable assignment is **first** used.

[1 mark]

.....

3	b		(line) 3;	1
3	c		(line) 4;	1
3	d		(line) 2;	1

Question and Mark Scheme from 4512 – June 2016

6 (e) What is the output from the procedure call `Display(3, (3-1))`?

[2 marks]

6	**e**		1 mark for starting at **3**; 1 mark for outputting **6** and nothing further (except 3);	**2**

10 (a) Tick the line of code that is equivalent to lines 6 **and** 7 together.

[1 mark]

Line of code	Tick **one** box
`IF player1 = 1 OR player2 = 2 THEN`	
`IF player1 ≠ player2 THEN`	
`IF player1 = 1 AND player2 = 2 THEN`	

10 (b) What data structure has been used for `options`?

[1 mark]

10 (c) Tick the programming technique that has **not** been used in this algorithm.

Programming technique	Tick **one** box
Iteration	
Selection	
Variable assignment	

[1 mark]

10 (d) Using either pseudocode or a flowchart, extend the algorithm in **Figure 5** so that the variable `draw` is set to the value `true` when both player 1 and player 2 choose the same option. This code should follow on from the end of the algorithm in **Figure 5**.

[3 marks]

10 (e) Using either pseudocode or a flowchart, extend the algorithm in **Figure 5** and your answer to question **10(d)** by outputting:

- either that the game was a draw
- or, if it was not a draw, which option beat which option.

Example 1: If player 1 entered a `2` and player 2 entered a `3` then the algorithm should output:

```
rock
beats
scissors
```

Example 2: If player 1 entered a `1` and player 2 entered a `2` then the algorithm should output:

```
rock
beats
paper
```

Example 3: If both players entered `1` then the algorithm should output:

```
draw
```

Your answer should extend the algorithm in **Figure 5** and leave the original algorithm unchanged.

Note: assume that array indexing starts at 1 so `OUTPUT options[1]` will output the value `paper`.

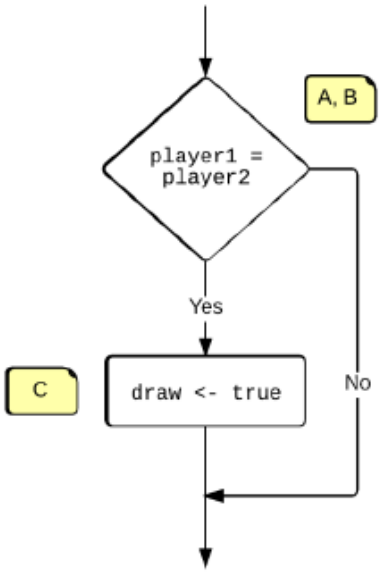
[9 marks]

<table>
<tr><td>10</td><td>a</td><td></td><td>Third box only;

<table><tr><th>Line of code</th><th>Tick one box</th></tr><tr><td>IF player1 = 1 OR player2 = 2 THEN</td><td></td></tr><tr><td>IF player1 ≠ player2 THEN</td><td></td></tr><tr><td>IF player1 = 1 AND player2 = 2 THEN</td><td>✓</td></tr></table></td><td>1</td></tr>
<tr><td>10</td><td>b</td><td></td><td>Array // list;

A. correct programming language-specific data structure
R. String</td><td>1</td></tr>
<tr><td>10</td><td>c</td><td></td><td>Correct row only;

<table><tr><th>Programming technique</th><th>Tick one box</th></tr><tr><td>Iteration</td><td>✓</td></tr><tr><td>Selection</td><td></td></tr><tr><td>Variable assignment</td><td></td></tr></table>
R. more than one row ticked</td><td>1</td></tr>
</table>

10	**d**		Marks awarded as follows (allow any logically equivalent and correct answer). The marks are labelled **A – C** and shown in the examples where they are awarded: **A.** 1 mark for using selection; **B.** 1 mark for a Boolean condition that tests for equivalency of the variables `player1` and `player2`; **C.** 1 mark for assigning the value `true` to the variable `draw`; **Example 1** (italicised square brackets indicate where marks are awarded): `IF player1 = player2 THEN` ***[A][B]*** `  draw ← true` ***[C]*** `ENDIF` **Example 2** where the inverse is tested in a Boolean condition (italicised square brackets indicate where marks are awarded): `draw ← true` ***[C]*** `IF player1 ≠ player2 THEN` ***[A][B]*** `  draw ← false` ***[also needed for C]*** `ENDIF` **Example 3** (notes indicate where marks are awarded) 	**3**

10	**e**		Marks awarded as follows (allow any logically equivalent and correct answer). The marks are labelled **A** – **I** and shown in the examples where they are awarded:
			
			A. 1 mark for using selection that 'divides' the code for a draw from the code for when it is not a draw. This would probably be either two `IF` statements or an `IF-ELSE`;
			B. 1 mark for the correct Boolean condition(s) with the selection statements in mark **A**;
			C. 1 mark for outputting `'draw'`;
			D. 1 mark if the output from mark **C** is within the correct part of the selection statement;
			E. 1 mark for using selection with the correct condition(s) to ascertain which player won (**I.** if this and subsequent lines of code are not within the correct part of the selection from mark **A**);
			F. 1 mark for ensuring the winning player's choice will output first (even if the output is incorrect);
			G. 1 mark for outputting the player's choice (even if this is not the winning player);
			H. 1 mark for outputting the string `'beats'`;
			I. 1 mark for outputting the choice of the other player from that used in mark **F**;
			
			Example 1 (italicised square brackets indicate where marks are awarded):
			
			`IF draw = true THEN` ***[A] [B]***
			`  OUTPUT 'draw'` ***[C] [D]***
			`ELSE`
			`  IF player1HasWon = true THEN` ***[E]***
			`  OUTPUT options[player1]` ***[F] [G]***
			`  ELSE`
			`  OUTPUT options[player2]` ***[I]***
			`  ENDIF`
			`  OUTPUT "beats"` ***[H]***
			`  IF player1HasWon = true THEN`
			`  OUTPUT options[player2]` ***[also needed for F]***
			`  ELSE`
			`  OUTPUT options[player1]` ***[also needed for I]***
			`  ENDIF`
			`ENDIF`
				9

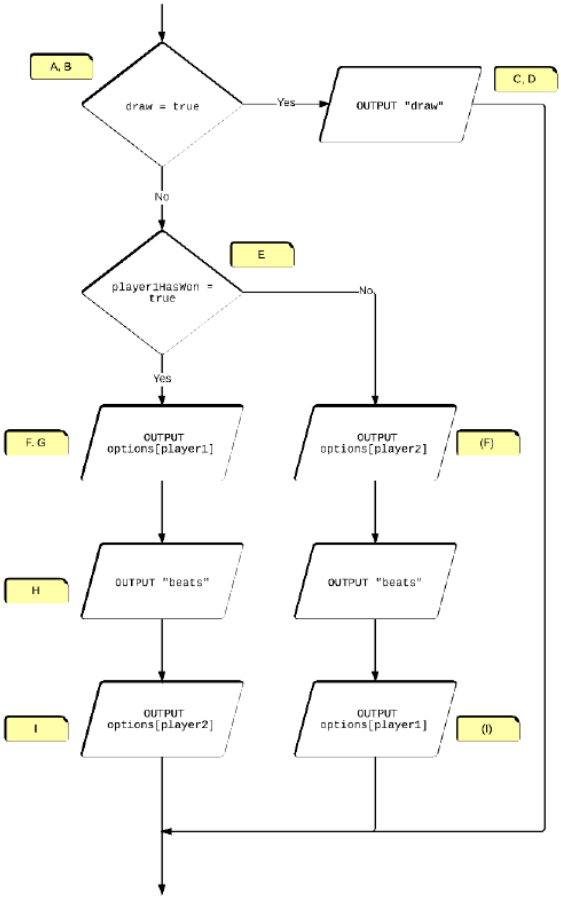
Example 2 (italicised square brackets indicate where marks are awarded):

```
IF draw = false THEN [A] [B]
  IF player1HasWon = true THEN [E]
    OUTPUT options[player1] [F] [G]
    OUTPUT "beats"  [H]
    OUTPUT options[player2] [I]
  ELSE
    OUTPUT options[player2] [also needed for F]
    OUTPUT "beats"  [also needed for H]
    OUTPUT options[player1] [also needed for I]
  ENDIF
ELSE
  OUTPUT "draw" [C] [D]
ENDIF
```

Example 3 (italicised square brackets indicate where marks are awarded):

```
IF player1HasWon = true THEN [A] [B] [E]
  OUTPUT options[player1] [F] [G]
  OUTPUT "beats" [H]
  OUTPUT options[player2] [I]
ENDIF
IF player1HasWon = false AND draw = false THEN
[also needed for A] [also needed for B] [also needed or E]
  OUTPUT options[player2] [also needed for F]
  OUTPUT "beats" [also needed for H]
  OUTPUT options[player1] [also needed for I]
ENDIF
IF draw = true THEN [also needed for A] [also needed
for B]
  OUTPUT "draw" [C] [D]
ENDIF
```

Example 4 (notes indicate where marks are awarded):


A, B
draw = true
Yes
OUTPUT "draw"
C, D
No
E
player1HasWon = true
No
Yes
F, G
OUTPUT options[player1]
OUTPUT options[player2]
(F)
H
OUTPUT "beats"
OUTPUT "beats"
I
OUTPUT options[player2]
OUTPUT options[player1]
(I)

Topic: 3.2.6 Data structures

Question and Mark Scheme from 4512 – June 2015

2 (a) Items of data can be combined together to form a data structure.

State the name of a programming language you are familiar with.

Programming language: .....

Give **one** example of a data structure that can be used in that programming language.

Data structure: .....

[1 mark]

2 (b) A programmer is developing a program that needs to record the names and ages of a group of students. Give **three** advantages of using a data structure to hold this information instead of using individual, separate variables for each name and age.

[3 marks]

Advantage 1.....

.....

.....

Advantage 2.....

.....

.....

Advantage 3.....

.....

.....

2	**a**		No marks for the programming language alone. Any correct combination of data structure and language. Examples include: • Python and list • Python and dictionary • Java and array • C and struct **A.** Any of array//list//record//struct//class data structures if given without naming a programming language.	**1**
2	**b**		Any creditworthy point to a maximum of three. Examples include: The number of students may be unknown; A data structure will be easier to iterate over/traverse; A data structure could hold both the names and ages together; A data structure would make the program code easier to update/modify; Could use pre-written routines with a standard data structure; Could make it easier to reuse the code; **A.** examples such as 'easier to sort the data'.	**3**

Topic: 3.2.10 Subroutines (procedures and functions)

Question and Mark Scheme from 4512 – June 2014

3 **Figure 1** shows a pseudocode representation of the function called `FindHighest`. `FindHighest` is used to find the largest value stored in an array.

Note: line numbers have been shown but are not part of the function.

Figure 1

```
1         FUNCTION FindHighest(arr)
2           highest ← arr[1]
3           FOR i ← 2 TO LEN(arr)
4             IF arr[i] > highest THEN
5               highest ← arr[i]
6             ENDIF
7           ENDFOR
8           RETURN highest
9         ENDFUNCTION
```

3 (a) How many parameters does the function `FindHighest` have?

[1 mark]

.....

3 (e) The variable `i` in **Figure 1** only has scope between lines `3` and `7`. Explain with reference to the variable `i` what scope means.

[1 mark]

.....

3	a		1;	1

3	e		The variable `i` can only be accessed/used/changed within those lines; The variable `i` is only defined within those lines; Trying to access the variable `i` outside of those lines will not work;	1

Question and Mark Scheme from 4512 – June 2015

7 The following function calculates the second hand price of different models of car. The parameter `condition` is an integer with a value between 1 and 4 where 1 is excellent and 4 is very bad.

```
FUNCTION CarPrice(model, condition, age)
  cost ← 0

  IF model = 'Daley' THEN
    cost ← 6000
  ELSE
    IF model = 'Minty' THEN
      cost ← 4000
    ELSE
      cost ← 2000
    ENDIF
  ENDIF

  CASE condition OF
    1: cost ← cost - 100
    2: cost ← cost - 300
    3: cost ← cost - 500
    4: cost ← cost - 1000
  ENDCASE

  cost ← cost / age

  RETURN cost
ENDFUNCTION
```

7 (a) `FUNCTION` and `ENDFUNCTION` are keywords in `CarPrice` that indicate it is a function. Which other keyword indicates that it is a function?

[1 mark]

.....

.....

7 (d) State **three** advantages of using functions/procedures when developing a program.

[3 marks]

Advantage 1.....

.....

Advantage 2.....

.....

Advantage 3.....

.....

7	**a**		RETURN; Do not penalise spelling mistakes as long as the word is clear.	**1**

7	**d**		1 mark for any correct answer to a maximum of 3. Answers include: It reduces repetition of code; It is easier to test; It is easier to maintain/update the program; It makes code more reusable; It makes code more elegant/understandable; It makes it easier for code to be developed in teams; It allows use of pre-written routines; It can speed up development time; **A.** Any other sensible answers.	**3**

Question and Mark Scheme from 4512 – June 2016

6 (d) Complete the trace table below showing the changes in the variable `x` and the output for the procedure call `Mult(2,3)`.

x	Output

[4 marks]

6 (e) What is the output from the procedure call `Display(3, (3-1))`?

[2 marks]

_____

_____

_____

_____

6 (f) State **two** reasons why writing your own functions/procedures in a program can make your code more reliable.

[2 marks]

1 _____

_____

2 _____

_____

<table>
<tr>
<td>6</td>
<td>d</td>
<td></td>
<td>1 mark for the first value of 1;
1 mark for incrementing each row by 1;
1 mark for the last value of 4;
1 mark for outputs being twice the value of x if at least two values of x are given (I. if x is incorrect)

The completed correct trace table is:

<table>
<tr><th>x</th><th>Output</th></tr>
<tr><td>1</td><td>2</td></tr>
<tr><td>2</td><td>4</td></tr>
<tr><td>3</td><td>6</td></tr>
<tr><td>4</td><td></td></tr>
</table>
</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>e</td>
<td></td>
<td>1 mark for starting at 3;
1 mark for outputting 6 and nothing further (except 3);</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>f</td>
<td></td>
<td>Any creditworthy points to a maximum of two. Examples include:

They can be tested in isolation;
They only need to be tested once;
They can be more easily updated;
They make code easier to understand (to a human);
Likely to reduce number of lines of code in a program;</td>
<td>2</td>
</tr>
</table>

Topic: 3.2.12 Robust and secure programming

Question and Mark Scheme from 4512 – June 2016

3 The pseudocode in **Figure 1** is written to make sure that the user enters a value within a given range.

Figure 1

```
inp ← USERINPUT
WHILE inp ≤ 0 OR inp ≥ 10
  OUTPUT "not in range"
  inp ← USERINPUT
ENDWHILE
```

3 (a) (i) Tick the set of test data that is the **most** appropriate to check that the code works as expected.

Test data	Tick **one** box
-1, 0, 9, 10	
0, 1, 10, 11	
-1, 0, 10, 11	
0, 1, 9, 10	

[1 mark]

3 (a) (ii) Why is the set of test data that you have chosen in **Question 3(a)(i)** likely to be enough to show that the code in **Figure 1** works as expected?

[1 mark]

_____

_____

3	**a**	**i**	Fourth box only; **Test data** / **Tick one box**: -1, 0, 9, 10 — 0, 1, 10, 11 — -1, 0, 10, 11 — 0, 1, 9, 10 ✓	**1**
3	**a**	**ii**	They test the boundaries; **A.** other wording that has equivalent meaning.	**1**

Topic: 3.3 Fundamentals of data representation
3.3.1 Number bases

Question and Mark Scheme from 4512 – June 2014

1 (c) Give **one** reason why programmers often use hexadecimal, instead of binary, to represent numbers.

[1 mark]

.....

.....

1	**c**		1 mark each for any correct answer. Examples include: Hexadecimal is easier (for humans) to read (than binary); Hexadecimal is easier to convert (to binary) than denary; Numbers are displayed in a more compact way (in hexadecimal than in binary); It is quicker to type in (hexadecimal numbers than binary numbers); It is more accurate to type in (hexadecimal numbers than binary numbers); **R.** anything that implies less memory is used.	**1**

Topic: 3.3.2 Converting between number bases

Question and Mark Scheme from 4512 – June 2014

1 (a) State the denary representation of the binary number 10010111

[1 mark]

.....

.....

.....

1 (b) State the hexadecimal representation of the denary number 125. You must show your working.

[2 marks]

.....

.....

.....

.....

1 (d) The ASCII character set uses seven bits to encode every character.

What is the total number of characters that can be encoded in ASCII?

[1 mark]

.....

.....

1	**a**		151;	**1**
1	**b**		7D; **If there is no hexadecimal answer then** do not reward any working; **If the answer given is 7D then** reward any attempt at working; **If the hexadecimal answer given is not 7D then** a maximum of 1 mark can be awarded for any of the following working out stages: • convert to binary 0111 1101 • convert each of their nibbles to hex **A**. If incorrect bit pattern is converted to its corresponding hex value • show division of 125 by 16 giving the quotient and remainder;	**2**

1	**d**		128 (characters) // 2^7 (characters);	**1**

Question and Mark Scheme from 4512 – June 2015

1 (a) State the **denary** representation of the binary number 10111010.

[1 mark]

.....

.....

1 (b) State the **hexadecimal** representation of the binary number 1110.

[1 mark]

.....

.....

1 (c) State the **denary** representation of the hexadecimal number 4C. You **must** show your working.

[2 marks]

.....

.....

.....

.....

1	**a**		186;	**1**
1	**b**		E;	**1**
1	**c**		76; If the answer given is 76 then reward any attempt at working; If the answer given is not 76 then a maximum of 1 mark can be awarded for any of the following working out stages: • Show multiplication of 4 by 16 and another number between 0 and 16 by 1 (i.e. allow C to be incorrectly converted to decimal). • Convert to binary 1001100 but then incorrectly converted to denary // convert to binary 01001100 but then incorrectly converted to denary. • Convert to a binary number other than 1001100, which must consist of more than 4 bits, but then convert this binary number to its correct decimal representation.	**2**

Question and Mark Scheme from 4512 – June 2016

1 (a) State the **binary** representation of the denary number 87.

[1 mark]

1 (b) State the **binary** representation of the hexadecimal number CE. You must show your working.

[2 marks]

1 (c) Place these **three** numbers into order of size (**1–3** where **1** is the largest and **3** is the smallest).

Number	Order (1–3)
The denary number 12	
The binary number 1110	
The hexadecimal number D	

[2 marks]

1 (d) What is the minimum number of bits needed to be able to represent any character from a character set that contains only the 26 lower-case letters of the alphabet?

[1 mark]

<table>
<tr><th>Qu</th><th>Part</th><th>Sub-part</th><th>Marking Guidance</th><th>Marks</th></tr>
<tr><td>1</td><td>a</td><td></td><td>101 0111;

I. Leading zeros</td><td>1</td></tr>
<tr><td>1</td><td>b</td><td></td><td>1100 1110;

If answer given is 11001110 then reward any attempt at working;
If the answer given is not 11001110 then a maximum of 1 mark can be awarded for any of the following working out stages:

• C or E (but not both) are converted to an incorrect binary representation but are then combined with the other correct representation. For example C is converted incorrectly to 1001 but E is converted correctly to 1110 and the answer given is 10011110;
• C is converted to a denary number other than 12 and/or E is converted to a denary number other than 14 but both of the denary numbers are correctly converted to binary.
• The candidate has attempted to multiply 16 by 12 and 1 by 14 but has then incorrectly converted the result into binary (through either an initial multiplication error or binary conversion error but not both).</td><td>2</td></tr>
<tr><td>1</td><td>c</td><td></td><td>1 mark for one correct row;
Both marks for all three correct rows;

<table><tr><th>Number</th><th>Order (1 – 3)</th></tr><tr><td>The denary number 12</td><td>3</td></tr><tr><td>The binary number 1110</td><td>1</td></tr><tr><td>The hexadecimal number D</td><td>2</td></tr></table>
R. if duplicate numbers have been used</td><td>2</td></tr>
<tr><td>1</td><td>d</td><td></td><td>5;</td><td>1</td></tr>
</table>

2 (d) How many bits does ASCII use to represent a single character?

[1 mark]

_____

_____

<table>
<tr><td>2</td><td>d</td><td></td><td>7;

A. 8-bits; (extended ASCII)</td><td>1</td></tr>
</table>

Topic: 3.3.3 Units of information

Question and Mark Scheme from 4512 – June 2015

1 (d) Place the following quantities in order of size (1 – 4, where 1 is the smallest and 4 is the largest).

Quantity	Order (1 – 4)
15 bits	
3 nibbles	
2 bytes	
1 kilobyte	

[3 marks]

1	**d**		1 mark if 1 number correct; 2 marks if 2 numbers correct; 3 marks if all 4 numbers correct; The correct order is: 2, 1, 3, 4 **Quantity** / **Order (1-4)** 15 bits – 2 3 nibbles – 1 2 bytes – 3 1 kilobyte – 4	**3**

Topic: 3.3.5 Character encoding

Question and Mark Scheme from 4512 – June 2015

1 (e) ASCII is a character-encoding system that uses seven bits to represent each character. Complete the table stating the binary representation of the character g.

Character	Binary Representation
f	110 0110
g	

[1 mark]

1	e		110 0111; **R**. if more than 7 bits used (eg 0110 0111)	1

Question and Mark Scheme from 4512 – June 2016

2 (d) How many bits does ASCII use to represent a single character?

[1 mark]

2	d		7; **A.** 8-bits; (extended ASCII)	1

Topic: 3.3.6 Representing images

Question and Mark Scheme from 4512 – June 2014

1 (f) Describe how a black and white image could be represented as a bitmap in binary.

[3 marks]

.....

.....

.....

.....

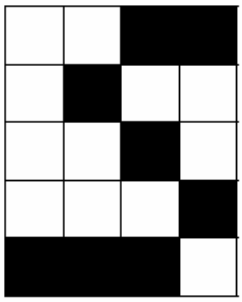
.....

.....

1	f		The image is represented as a series/grid/sequence of pixels; Each pixel/dot is represented by one bit; White is represented by a 0; Black is represented by a 1; **A.** White=1; Black=0; **A.** White and black are represented using different bit patterns (1 mark); **R.** Same bit pattern used for black and white Metadata about the image is also stored; A. examples of metadata **MAX 3**	3

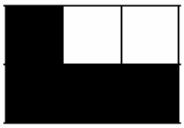
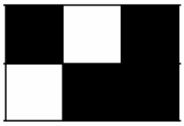
Question and Mark Scheme from 4512 – June 2015

1 (f) The following grid represents a bitmap image where a black pixel is represented using the bit pattern `00` and a white pixel is represented using the bit pattern `01`. The binary encoding of each row is shown next to the image.



```
01010000
01000101
01010001
01010100
00000001
```

1 (f) (i) Which **one** of the following images has the correct encoding?

	Image	Encoding	Tick one box
A		`010100` `000101`	☐
B		`00010100` `00000000`	☐
C		`000100` `010000`	☐

[1 mark]

1 (f) (ii) State the maximum number of different colours that can be encoded when using two bits for each pixel.

[1 mark]

.....

.....

1 (f) (iii) State the minimum number of bits needed to encode 32 different colours.

[1 mark]

.....

.....

1 (f) (iv) State **one** factor, other than the number of bits used to represent individual colours, that can affect the quality of a bitmap image.

[1 mark]

.....

.....

1	**f**	**i**	C; (correct answer only, do not award if more than one box is ticked)	**1**
1	**f**	**ii**	$4//2^2$;	**1**
1	**f**	**iii**	5;	**1**
1	**f**	**iv**	the resolution // number of pixels used // size of the grid // ppi (or equivalent) // compression;	**1**

Topic: 3.3.7 Representing sound

Question and Mark Scheme from 4512 – June 2014

1 (e) **Table 1** shows four stages in converting sound into a digital form.

Show the correct order for the stages by labelling them with the numbers 1 – 4 (1 being the first stage).

[3 marks]

Table 1

Stage	Order (1 – 4)
binary representation of level stored	
microphone picks up sound waves	
value read at specific point and rounded to a level	
converted to an electrical analogue signal	

1	**e**		1 mark if **1** stage correct 2 marks if **2** stages correct 3 marks if **all 4** stages correct The correct stages are: 4, 1, 3, 2	**3**

Topic: 3.3.8 Data compression

Question and Mark Scheme from 4512 – June 2014

1 (d) The ASCII character set uses seven bits to encode every character.

What is the total number of characters that can be encoded in ASCII?

[1 mark]

.....

.....

1	**d**		128 (characters) // 2^7 (characters);	**1**

Topic: 3.4 Computer systems
3.4.4 Systems architecture

Question and Mark Scheme from 4512 – June 2014

2 A typical computer's main memory consists of both volatile memory and non-volatile memory.

2 (a) (i) Explain what is meant by the term **volatile memory**.

[1 mark]

.....

.....

2 (a) (ii) What is normally stored in the non-volatile part of a computer's main memory?

[1 mark]

.....

.....

2 (b) Explain why having cache memory can improve the performance of the Central Processing Unit (CPU).

[2 marks]

.....

.....

.....

.....

2 (c) State **two** characteristics, other than the size of cache memory, that can improve the performance of CPUs.

[2 marks]

Characteristic 1 .....

.....

Characteristic 2 .....

.....

2	**a**	**i**	Memory content is lost when power is turned off; A. Any statement that implies temporary	**1**
2	**a**	**ii**	The computer's BIOS//initial instructions//bootstrapping instructions; **A.** Qualified answers about embedded systems eg washing machines. **A.** Operating system	**1**
2	**b**		Frequently used data/instructions are stored in the cache; Meaning they don't have to be fetched from main memory; Data/instructions stored in the cache memory can be accessed faster (than data/instructions stored in the main memory); **MAX 2**	**2**
2	**c**		1 mark each for any correct answer. Examples include: the number of cores/processors; the processing speed/clock speed/number of cycles (per second) of the processor; the bus width; the word size; the architecture of the processor/CPU; the type of cache memory; **R.** amount of cache memory **MAX 2**	**2**

5 **Figure 2** shows an example of a tablet computer.

Figure 2

© Thinkstock

5 (b) Tablet computers normally use solid state storage media instead of magnetic storage media.

State and explain **two** differences, other than cost and storage capacity, that make solid state media a better choice than magnetic media for tablet computers.

[4 marks]

Difference 1 .....

Explanation 1 .....

.....

.....

Difference 2 .....

Explanation 2 .....

.....

.....

5	**b**		Examples include: Difference: No mechanical parts in solid state media//Magnetic media has mechanical parts. Explanation: Magnetic media are often unsuitable for mobile use because the mechanical parts cannot function during movement // mechanical parts are less robust during movement. Difference: Speed of read access higher in solid state drives. Explanation: Data can often be read more quickly from solid state media than magnetic media. Difference: Solid state media can be more compact than magnetic media. Explanation: The smaller size enables better mobility; Reason: The battery will last longer Explanation: Solid state media uses less power Difference: Less heat generated when using solid state Explanation: Utilising the power more efficiently//allows for more miniaturisation. Difference: Solid state is silent Explanation: Makes it more attractive to use.	**4**

6 Explain how data is read from optical media such as a CD.

[5 marks]

6			1 mark for every correct point that explains the functionality of reading data from an optical medium such as a CD up to a maximum of 5 marks. Examples include: The tracking mechanism moves the laser into the correct position over the CD; The CD is spun to ensure all data can be read; The CD spins slower when the laser/read-head is above the outer tracks; The laser is shone on to the disk; The laser is reflected; Bumps/pits are raised parts of the disk; Bumps/pits form a spiral from the centre to the outside of the disk; A (opto-electric) sensor detects changes in reflectivity; Bumps/pits and lands represent the two possible bit values	**5**

Question and Mark Scheme from 4512 – June 2015

6 (a) What is a **computer system**?

[1 mark]

.....

.....

6 (b) Memory and the processor are two essential pieces of hardware. Explain, with reference to both memory and the processor, how a computer processes instructions.

[4 marks]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

6 (c) Give **one** reason why a CPU with two cores might perform faster than an equivalent CPU with only one core.

[1 mark]

.....

.....

.....

6 (d) The following are types of memory and storage (labelled A – F):

A. Cache memory
B. Magnetic media
C. Non-volatile memory
D. Optical media
E. ROM
F. Solid state media

For each of the descriptions in the table, write the label of the type of memory or storage it best describes.

Description	**Label (A – F)**
Uses a laser to read the data	
Contents cannot be edited	
Small and very fast storage found close to the processor	

[3 marks]

<table>
<tr><td>6</td><td>a</td><td></td><td>(A combination of) hardware and software;</td><td>1</td></tr>
<tr><td>6</td><td>b</td><td></td><td>One mark for each valid point below (maximum 4). If only one of memory or processor is referenced then maximum 3 marks.
The instructions are held in memory;
Loads instructions from secondary storage to memory;
Instructions are stored in a contiguous format;
The processor fetches an instruction from memory;
The processor decodes the instruction;
The processor executes the instruction;
The result may be stored back into memory;
The process is repeated continuously//cycles;

A. Any other correct answer</td><td>4</td></tr>
<tr><td>6</td><td>c</td><td></td><td>(Because the processor with two cores may be able to process) two instructions in parallel/at the same time/simultaneously;

A. Processing is shared.</td><td>1</td></tr>
<tr><td>6</td><td>d</td><td></td><td>The completed table is:
<table>
<tr><th>Description</th><th>Term</th></tr>
<tr><td>Uses a laser to read the data.</td><td>D (Optical media)</td></tr>
<tr><td>Contents cannot be edited.</td><td>E (ROM)</td></tr>
<tr><td>Small and very fast storage found close to the processor</td><td>A (Cache memory)</td></tr>
</table>
1 mark for each correct label.

A. The terms written out in full instead of the labels (do not penalise spelling errors)</td><td>3</td></tr>
</table>

Question and Mark Scheme from 4512 – June 2016

1 (f) Two typical secondary storage devices, with the same cost, are advertised as follows.

Device A	Device B
Solid state drive, capacity 128GB	Magnetic hard drive, capacity 1TB

1 (f) (i) State **one** reason why **Device B** could be considered a better choice than **Device A**.
[1 mark]

_____

_____

1 (f) (ii) State **two** reasons why **Device A** could be considered a better choice than **Device B**.
[2 marks]

_____

_____

_____

_____

1	f	i	It has a larger storage capacity / it can hold more data;	1
1	f	ii	Any creditworthy point to a **maximum of two**. Examples of typical advantages of solid state over magnetic storage include: It has a higher read/write speed; It is smaller; It is more robust; It generates less heat; It has a lower power consumption; It is lighter; It is quieter;	2

8 There are several CPU characteristics that can affect its performance. One of these is clock speed.

Explain how clock speed and **one** other CPU characteristic can affect CPU performance.

In this question you will be marked on your ability to use good English, to organise information clearly and to use specialist vocabulary where appropriate.

[6 marks]

8			No creditworthy material	0		**6**
			Lower mark range Vague statements are made about how clock speed and/or one other characteristic can affect CPU performance // Clock speed not mentioned but another CPU characteristic is described Quality of written communication: The candidate has used a form and style of writing which has many deficiencies. Ideas are not often clearly expressed. Sentences and paragraphs are often not well-connected or at times bullet points may have been used. Specialist vocabulary has been used inappropriately or not at all. Much of the text is	1-2 marks		

			legible and some of the meaning is clear. There are many errors of spelling, punctuation and grammar but it should still be possible to understand much of the response.			
			Mid mark range Clear descriptions are made about how clock speed affects performance. One other CPU characteristic is described. Quality of written communication: The candidate has mostly used a form and style of writing appropriate to purpose and has expressed some complex ideas reasonably clearly and fluently. The candidate has usually used well linked sentences and paragraphs. Specialist vocabulary has been used on a number of occasions but not always appropriately. Text is legible and most of the meaning is clear. There are occasional errors of spelling, punctuation and grammar.	3-4 marks		
			High mark range A correct and detailed explanation of how clock speed affects CPU performance is given, along with a correct and detailed description of one other CPU characteristic and its effect on performance. Quality of written communication: The candidate has selected and used a form and style of writing appropriate to purpose and has expressed complex ideas clearly and fluently. Sentences and paragraphs follow on from one another clearly and coherently. Specialist vocabulary has been used appropriately throughout. Text is legible and the meaning is clear. There are few if any errors of spelling, punctuation and grammar.	5-6 marks		
			Quality of written communication skills The candidate's quality of written communication skills will be one of the factors influencing the actual mark an examiner will give within a level of response. The quality of written communication skills associated with each level is indicated above.			

			Explanation of clock speed Instructions are fetched from memory; Decoded//Executed by the processor; The speed at which this cycle happens; Is directly related to the clock speed; So a higher clock speed means more instructions can be executed (per unit time). **Description of other characteristics may include:** **Cache memory** Frequently used instructions/data; Instructions/data which is predicted to be used; Are pre-loaded into cache; Which is faster to access than RAM/main memory; Is located on or close to the processor; Reduces the time to fetch data/instructions; **Number of cores** One processor/CPU has multiple cores; Each core can process instructions independently of the other; Allow more than one instruction/process to be processed in parallel;			