AQA

AS
COMPUTER SCIENCE

Paper 1

Monday 6 June 2016 Morning Time allowed: 1 hour 30 minutes

Materials
For this paper you must have access to:
- a computer
- a printer
- appropriate software
- the Electronic Answer Document
- an electronic version and a hard copy of the Skeleton Program
- an electronic version of the Data File
- an electronic version and a hard copy of the Preliminary Material.

You must **not** use a calculator.

Instructions
- Type the information required on the front of your Electronic Answer Document.
- Before the start of the examination make sure your **Centre Number**, **Candidate Name** and **Candidate Number** are shown clearly in the footer of every page of your Electronic Answer Document (not the front cover).
- Enter your answers into the Electronic Answer Document.
- Answer **all** questions.
- Save your work at regular intervals.

Information
- The marks for questions are shown in brackets.
- The maximum mark for this paper is 75.
- No extra time is allowed for printing and collating.
- The question paper is divided into **three** sections.

Advice
You are advised to allocate time to each section as follows:
Section A – 20 minutes; **Section B** – 20 minutes; **Section C** – 50 minutes.

At the end of the examination
Tie together all your printed Electronic Answer Document pages and hand them to the Invigilator.

Warning
It may not be possible to issue a result for this paper if your details are not on every page of your Electronic Answer Document.

Section A

You are advised to spend no more than **20 minutes** on this section.

Enter your answers to **Section A** in your Electronic Answer Document. You **must save** this document at regular intervals.

The questions in this section ask you to write program code **starting from a new program/project/file**.

You are advised to save your program at regular intervals.

---

0 1 In question parts **0 1 . 1** and **0 1 . 2** two statements are given followed by two conclusions numbered 1 and 2.

You must assume the two statements given in each question are true.

Read the statements and then decide which of the given conclusions logically follows from the two given statements.

Write the letter corresponding to your answer in your Electronic Answer Document.

0 1 . 1 Statements: All programmers work at night.
Nobody who works at night earns lots of money.

Conclusion 1: All programmers earn lots of money.
Conclusion 2: Some night workers are programmers.

Give answer:

A If only **Conclusion 1** follows.
B If only **Conclusion 2** follows.
C If neither **Conclusion 1** nor **Conclusion 2** follows.
D If both **Conclusion 1** and **Conclusion 2** follow.

[1 mark]

0 1 . 2 Statements: Some aardvarks are computing professors.
All computing professors love Java.

Conclusion 1: All aardvarks love Java.
Conclusion 2: All computing professors are aardvarks.

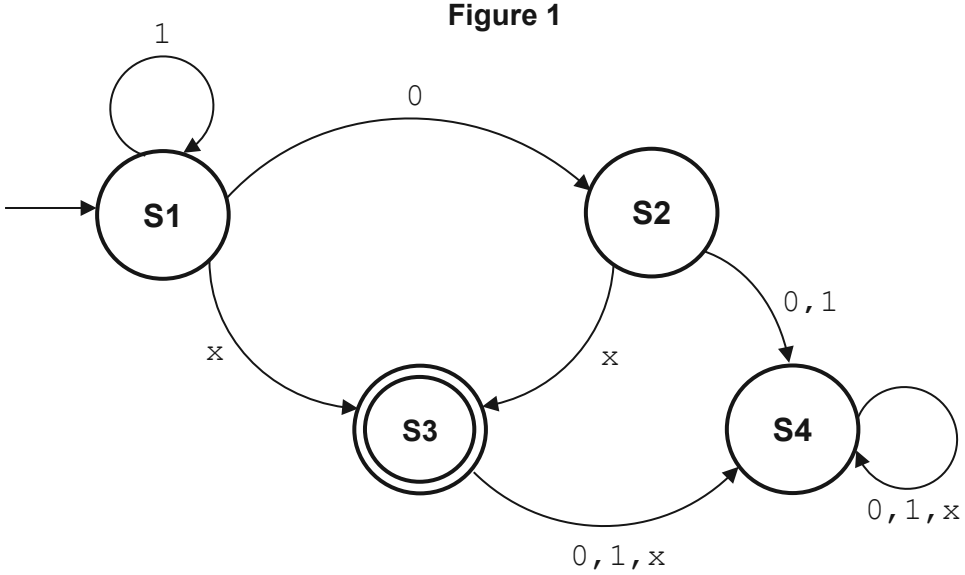
Give answer:

A If only **Conclusion 1** follows.
B If only **Conclusion 2** follows.
C If neither **Conclusion 1** nor **Conclusion 2** follows.
D If both **Conclusion 1** and **Conclusion 2** follow.

[1 mark]

0 2 The finite state machine (FSM) represented as a state transition diagram in **Figure 1** recognises a language with an alphabet of `0`, `1` and `x`.

Figure 1



Input strings of `0x` and `1x` are accepted by this FSM.

0 2 . 1 In **Table 1** indicate whether each input string is accepted or not accepted by the FSM in **Figure 1**.

If an input string is accepted write YES.
If an input string is **not** accepted write NO.

Complete **Table 1** by filling in the unshaded cells.

Copy the contents of all the unshaded cells in **Table 1** into your Electronic Answer Document.

Table 1

Input string	Accepted by FSM?
111011x	
1110x	
111001x	

[2 marks]

0 2 . 2 In words, describe the language (set of strings) that are accepted by the FSM in **Figure 1**.

[3 marks]

0 3 A new function `sqrt(x)` is being developed that returns the square root of a positive integer `x`.

There are three different types of test data.

Complete **Table 2** by stating the names of the **three different** types of test data that correspond to the values in the **Value for x** column that could be used when testing that the new `sqrt(x)` function works correctly.

Complete **Table 2** by filling in the unshaded cells.

Copy the contents of all the unshaded cells in **Table 2** into your Electronic Answer Document.

Table 2

Value for x	Type of test data
25	
1	
-8	

[2 marks]

There are no questions printed on this page

Turn over for the next question

0 4 The contents of the arrays `Items` and `NewItems` are shown in **Figure 2**.

A pseudo-code representation of an algorithm is given in **Figure 3**.

Figure 2

Items			
[0]	[1]	[2]	[3]
12	25	12	53

NewItems			
[0]	[1]	[2]	[3]
0	0	0	0

Figure 3

```
ItemsCount ← 4
NewItems[0] ← Items[0]
NewItemsCount ← 1
FOR LoopA ← 1 TO ItemsCount - 1
  Done ← False
  FOR LoopB ← 0 TO NewItemsCount - 1
    IF Items[LoopA] = NewItems[LoopB] THEN
      Done ← True
    ENDIF
  ENDFOR
  IF Done = False THEN
    NewItems[NewItemsCount] ← Items[LoopA]
    NewItemsCount ← NewItemsCount + 1
  ENDIF
ENDFOR
```

0 4 . 1 Dry run the algorithm in **Figure 3** by completing **Table 3**. The first row has been completed for you. You may not need to use all of the rows provided in the table.

Copy the contents of all the unshaded cells in **Table 3** into your Electronic Answer Document.

Table 3

ItemsCount	NewItemsCount	LoopA	Done	LoopB	NewItems			
					[0]	[1]	[2]	[3]
4	1				12	0	0	0

[5 marks]

0 4 . 2 Explain the purpose of the algorithm in **Figure 3**.

[1 mark]

Turn over for the next question

0 5 The algorithm, represented using pseudo-code in **Figure 4**, describes a method to calculate the additive or multiplicative persistence of a two-digit integer. The examples below illustrate how additive and multiplicative persistence are calculated.

Example: calculating the additive persistence of 87

8 + 7 = 15
1 + 5 = 6

After 2 steps the method results in a one digit answer so the additive persistence of 87 is 2.

Example: calculating the multiplicative persistence of 39

3 * 9 = 27
2 * 7 = 14
1 * 4 = 4

After 3 steps the method results in a one digit answer so the multiplicative persistence of 39 is 3.

Figure 4

```
OUTPUT "Enter integer (0-99): "
INPUT Value
OUTPUT "Calculate additive or multiplicative persistence (a or m)? "
INPUT Operation
Count ← 0
WHILE Value > 9
  IF Operation = "a" THEN
    Value ← (Value DIV 10) + (Value MOD 10)
  ELSE
    Value ← (Value DIV 10) * (Value MOD 10)
  ENDIF
  Count ← Count + 1
ENDWHILE
OUTPUT "The persistence is: "
OUTPUT Count
```

The `MOD` operator calculates the remainder resulting from an integer division, for example, `10 MOD 3 = 1`.

The `DIV` operator calculates integer division, for example `10 DIV 3 = 3`.

What you need to do

Task 1
Write a program for the algorithm in **Figure 4**.

Task 2
Test the program by showing the result of entering `47`, followed by `m` when prompted by the program.

Task 3
Test the program by showing the result of entering `77`, followed by `a` when prompted by the program.

Evidence that you need to provide
Include the following evidence in your Electronic Answer Document.

0 5 . 1 Your PROGRAM SOURCE CODE.

[8 marks]

0 5 . 2 SCREEN CAPTURE(S) showing the tests described.

[1 mark]

The part of the program where the calculations are performed uses a `WHILE` repetition structure.

0 5 . 3 Explain why a `WHILE` repetition structure was chosen instead of a `FOR` repetition structure.

[1 mark]

Turn over for Section B

Section B

You are advised to spend no more than **20 minutes** on this section.

Enter your answers to **Section B** in your Electronic Answer Document. You **must save** this document at regular intervals.

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

Refer **either** to the **Preliminary Material** issued with this question paper **or** your electronic copy.

0 6 State the name of an identifier for:

0 6 . 1 a variable that is used to store a single character.

[1 mark]

0 6 . 2 a user-defined subroutine that has one parameter.

[1 mark]

0 6 . 3 a user-defined subroutine that returns a Boolean value.

[1 mark]

0 7

0 7 . 1 A constant is used to hold the filename `'Training.txt'`.
State **one** advantage of using named constants for constant values.

[1 mark]

When validating the placement of a ship the `ValidateBoatPosition` subroutine is called.

0 7 . 2 Explain why a `FOR` loop is used as part of checking a valid vertical ship placement.

[2 marks]

0 7 . 3 Explain the purpose of the check that is being performed in the first selection structure of the `ValidateBoatPosition` subroutine.

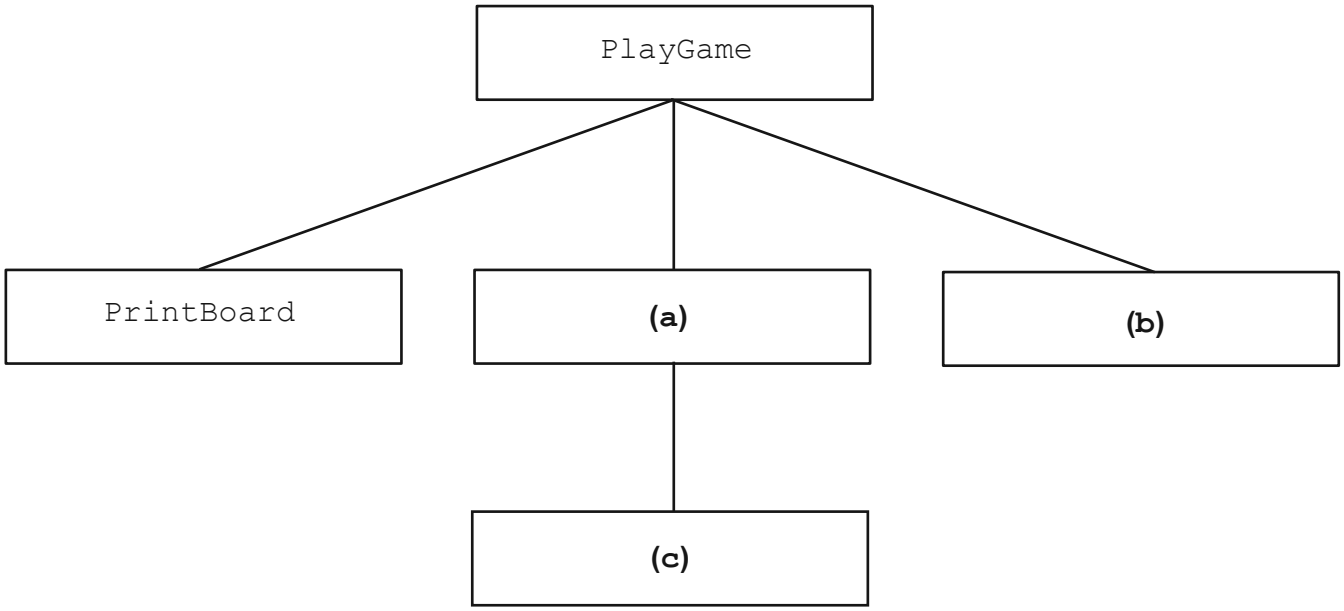
[2 marks]

0 7 . 4 Explain what is meant by exception handling **and** how exception handling could be used in the `GetRowColumn` subroutine.

[3 marks]

0 8 **Figure 5** shows an incomplete hierarchy chart for part of the **Skeleton Program**.

Figure 5



With reference to the **Skeleton Program** and **Figure 5**, answer the questions below.

0 8 . 1 What should be written in box **(a)** in **Figure 5**?

[1 mark]

0 8 . 2 What should be written in box **(b)** in **Figure 5**?

[1 mark]

0 8 . 3 What should be written in box **(c)** in **Figure 5**?

[1 mark]

A structured programming approach has been used in the production of the Skeleton Program.

0 8 . 4 Explain what is meant by a structured programming approach.

[2 marks]

Question 8 continues on the next page

There is a variable called `Row` in the subroutine `SetUpBoard`.
There is also a different variable called `Row` in the subroutine `LoadGame`.

0 8 . 5 Explain why these two different variables can have the same identifier.

[2 marks]

When the training game is selected from the main menu the positions of the ships are loaded from a text file. A binary file could have been used instead.

0 8 . 6 Describe a difference between the way in which data are stored in a binary file and the way data are stored in a text file.

[2 marks]

Section C

You are advised to spend no more than **50 minutes** on this section.

Enter your answers to **Section C** in your Electronic Answer Document. You **must save** this document at regular intervals.

These questions require you to load the **Skeleton Program** and to make programming changes to it.

0 9 This question refers to the subroutine `GetRowColumn`.

What you need to do:

Task 1
Adapt the program source code for the subroutine `GetRowColumn` so that it checks that the **row** entered by the player is in the allowed range.

If an invalid value is entered for the row the program should output:

```
Invalid value entered
```

The subroutine should not return any values until a valid row has been entered.

Task 2
Test that the changes you have made work by conducting the following test:

- run the **Skeleton Program**
- select option 2 from the menu
- fire a shot at column 6, row 10
- fire a shot at column 6, row 9.

Evidence that you need to provide
Include the following evidence in your Electronic Answer Document.

0 9 . 1 Your amended PROGRAM SOURCE CODE for the subroutine `GetRowColumn`.
[5 marks]

0 9 . 2 SCREEN CAPTURE(S) showing the requested test.
[1 mark]

1 0 This question will extend the functionality of the game.

The game needs to be changed so that a message is displayed when all the squares occupied by a ship have been hit.

Figure 6 shows an example of this functionality of the game using the board from the training game.

Figure 6

The player has made three shots and achieved 2 hits and a miss.

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9					m	h	h			

The player's next shot into column 7, row 9 is a third hit and sinks the ship.
The game displays the message `'Destroyer is sunk!'` to the player.

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9					m	h	h	h		

```
Destroyer is sunk!
```

What you need to do:

Task 1
Create a new subroutine `CheckSunk` that, when given the current shot location, board and list of ships, will:

- check the board to find out which kind of ship has been hit
- reduce, by one, the number of squares occupied by that ship in the list of ships
- display the message `'XX is sunk!'` where `XX` is the type of the ship, if the ship now occupies zero squares.

Task 2
Modify the `MakePlayerMove` subroutine so that the subroutine `CheckSunk` is called when appropriate.

Task 3
Test your program works by conducting the following test:

- run the **Skeleton Program**
- select option 2 from the menu
- fire a shot at column 1, row 3
- fire a shot at column 1, row 4
- fire a shot at column 1, row 5.

Evidence that you need to provide
Include the following evidence in your Electronic Answer Document.

1 0 . 1 Your PROGRAM SOURCE CODE for the subroutine `CheckSunk`.

[8 marks]

1 0 . 2 Your amended PROGRAM SOURCE CODE for the subroutine `MakePlayerMove`.

[1 mark]

1 0 . 3 SCREEN CAPTURE(S) showing the final board layout and the message `'Patrol Boat is sunk!'`.

[1 mark]

1 1 This question will further extend the functionality of the game.

The game is to be altered so that a player has the option of firing either a standard shot or a torpedo.

The player can only fire a torpedo once during the game.

If a player decides to fire a torpedo it behaves in the following manner:

a. if the torpedo lands directly on a ship (even if this part of the ship has already been hit) it behaves like a standard shot and explodes resulting in the square being marked as a hit and an appropriate message is displayed
b. if the torpedo lands in an empty square or a square already marked as a miss then the torpedo keeps moving up one square until it either hits a ship or moves off the board
c. each square that the torpedo has moved from will be marked as a miss
d. if the torpedo moves into a square that is occupied by a ship this will cause a hit like a standard shot (even if this part of the ship has already been hit), the torpedo will be removed from the board and an appropriate message is displayed
e. if the torpedo moves off the board it will disappear and an appropriate message is displayed.

You can choose to display the board each time the torpedo moves or to display the board only after the torpedo has finished moving.

Figures 7 to **9** show an example of a torpedo being used.

Figure 7

A torpedo is fired at column 8, row 6 and does not hit a ship.
At this stage no change is made to the contents of the board as the torpedo did not achieve a direct hit.

	0	1	2	3	4	5	6	7	8	9
0									m	
1									h	
2									h	
3										
4										
5										
6										
7										
8										
9										

Figure 8

The torpedo moves up one square but does not hit anything.
Column 8, row 6 (the square the torpedo came from) is marked as a miss.

	0	1	2	3	4	5	6	7	8	9
0									m	
1									h	
2									h	
3										
4										
5										
6									m	
7										
8										
9										

Figure 9

The torpedo moves up one square and hits a ship.
The torpedo is removed from the board.
Column 8, row 5 (the square the torpedo came from) is marked as a miss.
Column 8, row 4 is marked as a hit.

	0	1	2	3	4	5	6	7	8	9
0									m	
1									h	
2									h	
3										
4									h	
5									m	
6									m	
7										
8										
9										

Question 11 continues on the next page

What you need to do:

Task 1
Create a new subroutine `MakePlayerTorpedoMove` that when given a current board:

- asks the player to enter the row and column
- fires a torpedo starting at the square indicated by the player.

The torpedo should behave as described in **points a–e** on **page 16**.

Task 2
Adapt the `PlayGame` subroutine so that, if the player has not already fired a torpedo during the game, the player is asked:

```
Fire a torpedo? (Y/N)
```

If the player responds '`N`' a standard shot should be fired.

If the player responds '`Y`' the subroutine `MakePlayerTorpedoMove` should be called.

Task 3
Test your program works by conducting the following test:

- run the **Skeleton Program**
- select option 2 from the menu
- fire a torpedo at column 1, row 7
- fire a standard shot at column 2, row 1.

Evidence that you need to provide
Include the following evidence in your Electronic Answer Document.

1 1 . 1 Your amended PROGRAM SOURCE CODE for the subroutines `MakePlayerTorpedoMove` and `PlayGame` and any other code that you have changed or added.

[12 marks]

1 1 . 2 SCREEN CAPTURE(S) showing the requested test.

[2 marks]

END OF QUESTIONS

There are no questions printed on this page

There are no questions printed on this page