AQA

AS
COMPUTER SCIENCE

Paper 1

Date Morning 1 hour 45 minutes

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

Instructions
- Type the information required on the front of your **Electronic Answer Document**.
- Enter your answers into the **Electronic Answer Document**.
- Answer **all** questions.
- 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).
- Tie together all your printed **Electronic Answer Document** pages and hand them to the invigilator.

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.
- You are **not** allowed to use a calculator.

Advice
- You are advised to spend time on each section as follows:
  Section A – 20 minutes
  Section B – 20 minutes
  Section C – 65 minutes.
- Save your work at regular intervals.

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 question in this section asks you to write program code **starting from a new program/project/file.**

- Save your program/project/file in its own folder/directory.
- You are advised to save your program at regular intervals.

0 1 **Figure 1** contains the pseudo-code for a program to output a sequence according to the 'Fizz Buzz' counting game.

Figure 1

```
OUTPUT "How far to count?"
INPUT HowFar
WHILE HowFar < 1
  OUTPUT "Not a valid number, please try again."
  INPUT HowFar
ENDWHILE
FOR MyLoop ← 1 TO HowFar
  IF MyLoop MOD 3 = 0 AND MyLoop MOD 5 = 0
  THEN
      OUTPUT "FizzBuzz"
    ELSE
      IF MyLoop MOD 3 = 0
        THEN
          OUTPUT "Fizz"
        ELSE
          IF MyLoop MOD 5 = 0
            THEN
              OUTPUT "Buzz"
            ELSE
              OUTPUT MyLoop
          ENDIF
      ENDIF
  ENDIF
ENDFOR
```

What you need to do:

Write a program that implements the pseudo-code as shown in **Figure 1.**

Test the program by showing the result of entering a value of `18` when prompted by the program.

Test the program by showing the result of entering a value of `-1` when prompted by the program.

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

0 1 . 1 Your PROGRAM SOURCE CODE for the pseudo-code in **Figure 1**.

[8 marks]

0 1 . 2 SCREEN CAPTURE(S) for the tests conducted when a value of `18` is entered by the user and when a value of `-1` is entered by the user.

[1 mark]

The main part of the program uses a `FOR` repetition structure.

0 1 . 3 Explain why a `FOR` repetition structure was chosen instead of a `WHILE` repetition structure.

[1 mark]

Even though a check has been performed to make sure that the variable `HowFar` is greater than `1` there could be inputs that might cause the program to terminate unexpectedly (crash).

0 1 . 4 Provide an example of an input that might cause the program to terminate and describe a method that could be used to prevent this.

[3 marks]

Question 1 continues on the next page

Turn over

Programs written in a high level language are easier to understand and maintain than programs written in a low level language.

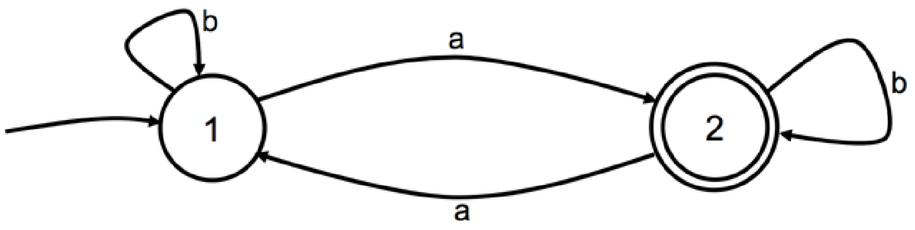
The use of meaningful identifier names is one way in which high level languages can be made easier to understand.

0 1 . 5 State **three** other features of high level languages that can make high level language programs easier to understand.

[3 marks]

The finite state machine (FSM) shown in **Figure 2** recognises a language with an alphabet of a and b.

Figure 2



Input strings of a and aabba would be accepted by this FSM.

0 1 . 6 In **Table 1** indicate whether each input string would be accepted or not accepted by the FSM in **Figure 2**.

If an input string would be accepted write YES.
If an input string would **not** be accepted write NO.

Copy your answer in **Table 1** into the Electronic Answer Document.

Table 1

Input string	Accepted by FSM?
aaab	
abbab	
bbbbba	

[2 marks]

0 1 . 7 In words, describe the language (set of strings) that would be accepted by this FSM shown in **Figure 2**.

[2 marks]

END OF SECTION A

There are no questions printed on this page

Turn over for Section B

Turn over

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 2 State the name of an identifier for:

0 2 . 1 an array or list variable

[1 mark]

0 2 . 2 a user-defined subroutine that has four parameters

[1 mark]

0 2 . 3 a variable that is used to store a whole number.

[1 mark]

0 2 . 4 a user-defined subroutine that returns one or more values.

[1 mark]

Look at the repetition structures in the `DisplayCavern` subroutine.

0 2 . 5 Explain the need for a nested `FOR` loop and the role of the `Count1` and `Count2` variables.

[3 marks]

Look at the `ResetCavern` subroutine.

0 2 . 6 Why has a named constant been used instead of the numeric value `5`?

[2 marks]

Look at the `SetPositionOfItem` subroutine.

0 2 . 7 Describe the purpose of the `WHILE` loop and the command within it in this subroutine.

[3 marks]

Look at the `MakeMonsterMove` subroutine.

0 2 . 8 Describe why it is necessary to check if the monster moves into the same cell as the flask and how any problem caused by this is solved by the **Skeleton Program**.

[3 marks]

Look at the `PlayGame` subroutine.

0 2 . 9 Explain why a `WHILE` loop has been made to complete the two moves for the monster rather than a `FOR` loop.

[2 marks]

The subroutines in the **Skeleton Program** avoid the use of global variables: they use local variables and parameter passing instead.

0 2 . 1 0 State **two** reasons why subroutines should, ideally, **not** use global variables.

[2 marks]

END OF SECTION B

Turn to page 9 for Section C

Turn over

There are no questions printed on this page

Section C

You are advised to spend no more than **65 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 3 This question refers to the subroutines `CheckValidMove` and `PlayGame`.

The **Skeleton Program** currently does not make all the checks needed to ensure that the move entered by a player is an allowed move. It should **not** be possible to make a move that takes a player outside the 7x5 cavern grid.

The **Skeleton Program** is to be adapted so that it prevents a player from moving north if they are at the northern end of the cavern.

The subroutine `CheckValidMove` needs to be adapted so that it returns a value of `False` if a player attempts to move north when they are at the northern end of the cavern.

The subroutine `PlayGame` is to be adapted so that it displays an error message to the user if an illegal move is entered. The message should state `"That is not a valid move, please try again."`.

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

0 3 . 1 Your amended PROGRAM SOURCE CODE for the subroutine `CheckValidMove.`

[4 marks]

0 3 . 2 Your amended PROGRAM SOURCE CODE for the subroutine `PlayGame.`

[1 mark]

0 3 . 3 SCREEN CAPTURE(S) for a test run showing a player trying to move north when they are at the northern end of the cavern.

[1 mark]

0 4 This question refers to the `PlayGame` subroutine and will extend the functionality of the game.

A scoring system is to be implemented as a game of MONSTER! is played. A variable called `Score` will be used to store the current score of each player.

The final score will be displayed to the user at the end of the game. At the end of the game, either the player will have found the flask or the player will have been eaten by the monster.

The final score should be displayed with the message `"Your score was: Y"` where Y is the value of `Score`.

The scoring system will be based upon the following:

- each valid move by the player is +10 points
- finding the flask is +50 points
- setting off a trap is -10 points
- being killed by the monster is -50 points.

Task 1
Adapt the **Skeleton Program** so that the scoring system described above is implemented, with the value of Score being updated as indicated and the required message being displayed at the end of a game.

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

- play the training game
- move south
- move south
- move east.

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

0 4 . 1 Your amended PROGRAM SOURCE CODE for the subroutine `PlayGame` and (if relevant) the PROGRAM SOURCE CODE for any other subroutine(s) you have amended.

[8 marks]

0 4 . 2 SCREEN CAPTURE(S) showing the required test.

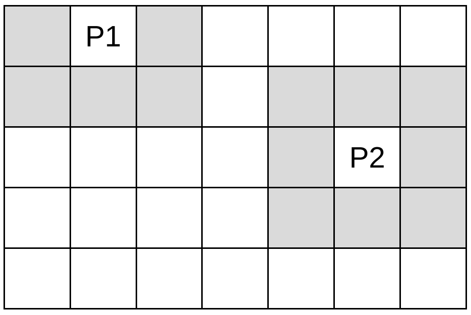
[1 mark]

0 5 This question will extend the functionality of the game.

The player will now have access to a close-range trap detector. After making a directional move in the cavern, the trap detector will perform a sweep of the neighbouring cells and report back if a trap is detected. Unfortunately, the detector can only detect the presence of a trap in a neighbouring cell, and not which individual cell the trap is in.

In **Figure 3** the shaded cells show the cells that would be scanned by the trap detector if the player were in the cell marked P1 or P2. The trap detector cannot scan outside the cavern.

Figure 3



Task 1
Create a new subroutine, `TrapDetector`, that, when given the current location of the player, returns `True` if a trap is in a neighbouring cell and `False` if there is no trap in a neighbouring cell.

When creating this subroutine you should ensure that your solution is efficiently coded.

Task 2
Modify the `PlayGame` subroutine so that after the player moves and the new state of the cavern is displayed:

- the message `‘Trap detected’` is displayed if there is a trap in any neighbouring cell.
- the message `‘No trap detected’` is displayed if there are no traps in any neighbouring cell.

Task 3
Test that your program works by loading the training game and showing that:

- a trap is detected after the player’s first move, west
- a trap is detected after the player’s second move, south
- a trap is not detected after the player’s third move, west**.**

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

0 5 . 1 Your PROGRAM SOURCE CODE for the subroutine `TrapDetector`.

[12 marks]

0 5 . 2 Your amended PROGRAM SOURCE CODE for the subroutine `PlayGame`.

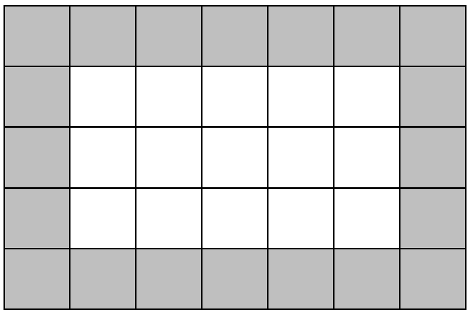
[1 mark]

0 5 . 3 SCREEN CAPTURE(S) showing the required sequence of tests being carried out, with the trap detected message being displayed after each of the first two moves and the trap not detected message being displayed after the third move.

[1 mark]

The game of MONSTER!, as represented by the **Skeleton Program**, is to be extended so that the cavern generated is not rectangular. The outer cells, shaded in **Figure 4**, will be randomly selected to be either rock or normal space when a new game starts. A cell that contains rock cannot be entered by the monster or player.

Figure 4



0 5 . 4 Describe changes that could be made to the **Skeleton Program** to achieve this.

In your answer you should ensure that you discuss changes to the data held in the `Cavern` variable and how the subroutines `ResetCavern` and `CheckValidMove` will need to be altered.

You are **not** expected to actually make the changes.

[5 marks]

0 5 . 5 A request has been made that the layout of the whole cavern should be more random. It has been suggested that all of the cells should be made a random choice between rock and normal space during setup.

Identify **two** problems that might occur with the MONSTER! game if this suggestion was made to the program.

[2 marks]

END OF QUESTIONS

There are no questions printed on this page

DO NOT WRITE ON THIS PAGE
ANSWER IN THE SPACES PROVIDED

There are no questions printed on this page

DO NOT WRITE ON THIS PAGE
ANSWER IN THE SPACES PROVIDED

Acknowledgement of copyright holders and publishers

Permission to reproduce all copyright material has been applied for. In some cases, efforts to contact copyright holders have been unsuccessful and AQA will be happy to rectify any omissions of acknowledgements in future papers if notified.

Copyright © 2014 AQA and its licensors. All rights reserved.