AQA

AS
COMPUTER SCIENCE

Paper 1

Tuesday 14 May 2024 Afternoon Time allowed: 1 hour 45 minutes

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

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 (not the front cover) of your Electronic Answer Document.
- 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** – 25 minutes; **Section C** – 60 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.

Question **04** in this section asks you to write program code **starting from a new program/project/file**.

You are advised to save your program at regular intervals.

0 1

Figure 1

```
S1 ← "011101"
S2 ← "001100"
C ← "0"
R ← ""
FOR J = 0 TO 5
  X ← 5 - J
  D1 ← S1[X]
  D2 ← S2[X]
  IF C = "0" THEN
    IF D1 = D2 THEN
      S ← "0"
      C ← D1
    ELSE
      S ← "1"
    ENDIF
  ELSE
    IF D1 = D2 THEN
      S ← "1"
      C ← D1
    ELSE
      S ← "0"
    ENDIF
  ENDIF
  R ← CONCATENATE(S, R)
ENDFOR
OUTPUT R
```

The function `CONCATENATE(X, Y)` returns the string formed by concatenating the string `Y` to the end of string `X`. For example, `CONCATENATE("cat", "dog")` returns `"catdog"`.

The strings are zero index based.

Complete **Table 1** by hand-tracing the algorithm in **Figure 1**.

You may not need to use all the rows in **Table 1**.

You do not need to indicate that `C`, `D1`, `D2` and `S` are strings.

The first row of **Table 1** has already been completed for you.

Table 1

S1	S2	C	R	J	X	D1	D2	S
"011101"	"001100"	"0"	""					
				OUTPUT:				

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

[5 marks]

0 2 A program uses both local and global variables.

0 2 . 1 State **two** differences between local and global variables.

[2 marks]

0 2 . 2 Give **two** reasons why it is good practice to use local variables.

[2 marks]

0 3 Programmers are encouraged to adopt a structured approach to writing programs.

Explain **three** reasons for adopting the structured approach.

[3 marks]

0 4 **Figure 2** shows an algorithm represented using pseudo-code.

Figure 2

```
OUT_NOLF "Enter an integer: "
INPUT Number1
OUT_NOLF "Enter another integer: "
INPUT Number2
IF Number1 > Number2 THEN
  Number ← Number1 DIV Number2
ELSE
  Number ← Number2 DIV Number1
ENDIF
Count ← 0
WHILE Count ≠ Number
  Count ← Count + 1
  IF (Count MOD 10) = 0 THEN
    OUT_NOLF "X"
  ELSE
    IF (Count MOD 5) = 0 THEN
      OUT_NOLF "V"
    ELSE
      OUT_NOLF "/"
    ENDIF
  ENDIF
ENDWHILE
```

The `OUT_NOLF` command displays the output without a line feed. The following series of `OUT_NOLF` commands will display `ABC`:

```
OUT_NOLF "A"
OUT_NOLF "B"
OUT_NOLF "C"
```

Table 2 lists the `MOD` and `DIV` operators for each of the available programming languages. You should refer to the row for your programming language.

Table 2

Programming language	MOD	DIV
C#	`%`	`/`
Java	`%`	`/`
Pascal	`mod`	`div`
Python	`%`	`//`
VB.NET	`Mod`	`\`

What you need to do:

Task 1
Write a program to implement the algorithm in **Figure 2**.

Task 2
Test that your program works:
- run your program
- enter `4`
- enter `99`

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

0 4 . 1 Your PROGRAM SOURCE CODE for **Task 1**.

[8 marks]

0 4 . 2 SCREEN CAPTURE(S) showing the test described in **Task 2**.

[1 mark]

Section B

You are advised to spend no more than **25 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 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 5 . 1 State the most appropriate data type to use for numbers with a fractional part, for example 12.79.

[1 mark]

0 5 . 2 State the identifier of a variable in the **Skeleton Program** that is used to hold a number with a fractional part.

[1 mark]

0 6 . 1 State the most appropriate data type to use for values that may consist of more than one character.

[1 mark]

0 6 . 2 State the identifier of a variable in the **Skeleton Program** that is used to hold more than one character.

[1 mark]

0 7 State the name of an identifier for a subroutine defined in the **Skeleton Program** that returns a single digit number.

[1 mark]

0 8 This question refers to the subroutine `CalculateServingTime`.

Explain the reason for adding the value `1` to the result of dividing `NoOfItems` by `TILL_SPEED`.

[1 mark]

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

Why might the subroutine `Serving` be called repeatedly **after** the specified time for the simulation has finished?

[2 marks]

1 0 The **Skeleton Program** uses a number of data structures.

1 0 . 1 State the identifier of a data structure that stores values of a **user-defined** data type.

[1 mark]

1 0 . 2 State the identifier of a data structure that stores values of only one **built-in** data type.

[1 mark]

1 1 . 1 Explain what is meant by composition **and** give an example where composition is used in the **Skeleton Program**.

[2 marks]

1 1 . 2 Describe **two** reasons why composition is used in the **Skeleton Program**.

[2 marks]

1 2 What is the named constant `TIME_SERVING` used for and what is the benefit of using a named constant in this way?

[2 marks]

1 3 This question refers to the subroutine `ServeBuyer`.

1 3 . 1 Describe the purpose of the statements within the FOR loop.

[2 marks]

1 3 . 2 What aspect of a shop do the FOR loop and the statements inside it simulate?

[1 mark]

1 4 One of the statistics calculated is average queue length.

This is calculated using the data structure `Stats`.

1 4 . 1 Describe how some of the values in `Stats` are used to calculate the average queue length in the subroutine `OutputStats`.

[1 mark]

1 4 . 2 Describe how the relevant values in `Stats` are updated in the subroutine `Serving` to enable the calculation in the subroutine `OutputStats` to produce the correct result.

[2 marks]

1 5 In the **Skeleton Program** buyers join a single queue and are then served at one of several tills.

Outline the design changes needed for buyers to form a separate queue for each till.

You are not expected to actually make the changes.

[2 marks]

Section C

You are advised to spend no more than **60 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.

1 6 This question extends the functionality of the **Skeleton Program**.

When a buyer enters the shop and sees a long queue, they will change their mind and leave the shop without buying anything. This is known as a **shun**.

The **Skeleton Program** is to be changed so that a buyer shunning the queue will cause the shop to open another till, subject to the maximum number of tills available not being exceeded.

The subroutine `BuyerArrives` needs to be modified so that a buyer arriving when the queue length is 5 will not join the queue. A running total of the number of buyers shunning the queue is to be kept.

After a buyer has shunned the queue, the shop will open another till, until the maximum number of tills available have opened.

At the end of the simulation the number of buyers that have shunned the queue should be output as part of the statistics.

What you need to do:

Task 1
Create a constant value to use as an index for a previously unused element of the data structure `Stats` in which to store the count of the number of shuns.

Task 2
Amend the subroutine `BuyerArrives` so that a buyer only joins the queue when fewer than 5 buyers are in the queue and shuns the queue if there are 5 buyers in the queue.

If a buyer shuns the queue, the number of shuns and the number of tills operating should be updated as appropriate and a suitable message showing the ID of the buyer that has shunned the queue should be displayed.

Task 3
Amend the subroutine `OutputStats` to output the total number of shuns with an appropriate message.

Task 4

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

- run your amended **Skeleton Program**
- enter `Y`
- enter `50`
- enter `1`

Evidence that you need to provide

Include the following evidence in your Electronic Answer Document.

1 6 . 1 Your PROGRAM SOURCE CODE:

- that sets the value of the new constant
- for the entire subroutine `BuyerArrives`
- for the entire subroutine `OutputStats`.

[7 marks]

1 6 . 2 SCREEN CAPTURE(S) showing the requested test described in **Task 4**.

The SCREEN CAPTURE(S) **only** need to show the simulation statistics that are displayed when the simulation finishes.

[1 mark]

Turn over for the next question

1 7 This question extends the functionality of the **Skeleton Program**, so that the length of time it takes to serve a buyer depends on the speed of the till operator. The speed of the till operator is the number of items they can handle in a time unit.

The simulation is to calculate the serving time for a buyer based upon the number of items in the buyer's basket and the speed of the till operator. The speed of each till operator needs to be stored for each till.

What you need to do:

Task 1
An extra value is required for each element of `Tills`, representing the speed of the operator at a till. Amend the declaration for `Tills`.

Task 2
Amend the subroutine `ResetDataStructures` so that the extra values are set so that till 0 has an operator speed of 7 and till 1 has an operator speed of 6. Each subsequent till should have its operator speed set to a value 1 less than the previous till.

Task 3
Amend the subroutine `ChangeSettings` so that it requires the user to set the till speed of each till in use. The user should be given information about which till they are setting the current till operator speed of and the current default value.
`ChangeSettings` will require `Tills` as a parameter and, depending on your language, as a return value, so the call to this subroutine will also need amending.

Task 4
Amend the subroutine `CalculateServingTime` so that the serving time depends on the speed of the till operator at the till that is being used.

Task 5
Test that the changes you have made work by conducting the following test:
- run your amended **Skeleton Program**
- enter `Y`
- enter `6`
- enter `3`
- enter `3`
- enter `2`
- enter `1`

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

1 7 . 1 Your PROGRAM SOURCE CODE for the entire subroutine `ResetDataStructures`, the entire subroutine `ChangeSettings` and the entire subroutine `CalculateServingTime` and any code that you have changed or added to the Skeleton Program.

[8 marks]

1 7 . 2 SCREEN CAPTURE(S) showing the requested test described in **Task 5**.

The SCREEN CAPTURE(S) **only** need to show the entire output for **time unit 5**.

[1 mark]

Turn over for the next question

1 8 This question extends the functionality of the **Skeleton Program**.

Till 0 is not currently used. It is to be used as the express till that buyers with fewer than 10 items in their basket can use.

If till 0 is idle, the next buyer in the queue with fewer than 10 items in their basket comes out of the queue and gets served at till 0.

What you need to do:

Task 1
Write a new subroutine `ServeBuyerExpress` that:

- finds the first buyer in the queue who has fewer than ten items in their basket and **if there is one**:
  - removes the buyer's details from the queue
  - closes the gap in the queue
  - displays the buyer's ID
  - calls the `UpdateStats` subroutine for the express till
  - calls the `CalculateServingTime` subroutine for the express till.

Task 2
Amend the subroutine `Serving` to check whether the express till is free and, if it is, call the subroutine `ServeBuyerExpress` before checking the availability of other tills.

Task 3
Amend the subroutine `OutputTillAndQueueStates` to output the data for till 0 as well as the data about the other tills which it already outputs.

Task 4
Test that the changes you have made work by conducting the following test:

- run your amended **Skeleton Program**
- enter `N`

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

1 8 . 1 Your PROGRAM SOURCE CODE for the entire subroutine `ServeBuyerExpress`, the entire subroutine `Serving` and the entire subroutine `OutputTillAndQueueStates`.

[12 marks]

1 8 . 2 SCREEN CAPTURE(S) showing the requested test described in **Task 4**.

The SCREEN CAPTURE(S) **only** need to show the entire output for **time unit 8**.

[1 mark]

END OF QUESTIONS

There are no questions printed on this page

There are no questions printed on this page

There are no questions printed on this page

There are no questions printed on this page

Copyright information

For confidentiality purposes, all acknowledgements of third-party copyright material are published in a separate booklet. This booklet is published after each live examination series and is available for free download from www.aqa.org.uk.

Permission to reproduce all copyright material has been applied for. In some cases, efforts to contact copyright-holders may have been unsuccessful and AQA will be happy to rectify any omissions of acknowledgements. If you have any queries please contact the Copyright Team.

Copyright © 2024 AQA and its licensors. All rights reserved.