

A



AS

COMPUTER SCIENCE

Paper 1

7516/1

Time allowed: 1 hour 45 minutes

[Turn over]

For this paper you must have:

- **a separate insert**
- **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 Files puzzle1.txt, puzzle1P.txt and puzzle1S.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.**

[Turn over]

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.

DO NOT TURN OVER UNTIL TOLD TO DO SO

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 03 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
---	---

The algorithm, represented using pseudo-code in FIGURE 1, on page 2 of the insert, describes a method to access numbers in the data structure, `List`, shown in TABLE 1, on page 8.

Complete TABLE 2, on page 9, by hand-tracing the algorithm in FIGURE 1 when the following statement is executed.

`Result ← A(38, 0, 18)`

You may not need to use all the rows in TABLE 2.

The first row of TABLE 2 has already been completed for you.

[Turn over]

TABLE 1**List**

[0]	2	[10]	57
[1]	8	[11]	61
[2]	12	[12]	68
[3]	18	[13]	71
[4]	25	[14]	79
[5]	29	[15]	83
[6]	36	[16]	84
[7]	42		
[8]	49		
[9]	51		

TABLE 2

S	X	Y	P	Z	List[Z]
38	0	18	-1		
Result:					

Copy the contents of all the unshaded cells in TABLE 2 into your Electronic Answer Document. [5 marks]

[Turn over]

0	2
---	---

The information for question 02 and FIGURE 2 are on pages 4 to 7, of the insert. Read this information before answering the question.

Complete TABLE 3, on pages 12 to 13, by filling in the unshaded cells with the correct labels from FIGURE 2. You should write:

- which labels (A) to (I) represent which event(s)**
- which labels (W) to (Z) represent which state.**

Some of the cells in the table may need to be assigned more than one label.

Each label MUST only be used once.

BLANK PAGE

[Turn over]

TABLE 3

Event / State	Label(s): (A) to (I), (W) to (Z)
Card Payment Mode	
Enter correct PIN	
Enter incorrect PIN	
Insert a coin (except final coin)	
Insert final coin	
Insert payment card	
Paid Mode	
Payment Due Mode	
Press Accept	

Press Cancel	
Press + button	
Select Hours Mode	

Copy the contents of all the unshaded cells in TABLE 3 into your Electronic Answer Document. [6 marks]

[Turn over]

03

FIGURE 3, on pages 8 and 9 of the insert, shows an algorithm represented using pseudo-code.

The `DIV` operator calculates the whole number part resulting from an integer division, for example, $10 \text{ DIV } 3 = 3$

TABLE 4 lists the `DIV` operators for each of the available programming languages. You should refer to the row for your programming language.

TABLE 4

Programming language	DIV
C#	/
Java	/
Pascal	div
Python	//
VB.NET	\

WHAT YOU NEED TO DO:

TASK 1

Write a program to implement the algorithm in FIGURE 3.

TASK 2

**Test that your program works:
run your program.**

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

03.1

Your PROGRAM SOURCE CODE for TASK 1. [8 marks]

03.2

SCREEN CAPTURE(S) showing the test described in TASK 2. [1 mark]

[Turn over]

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 4 . 1

State the most appropriate data type to use for whole numbers. [1 mark]

0	4	.	2
---	---	---	---

State the identifier of a variable in the SKELETON PROGRAM that can only represent whole numbers. [1 mark]

0	5
---	---

State the identifier AND data type of a variable in the SKELETON PROGRAM that can only represent two different values. [2 marks]

[Turn over]

0	6
---	---

The SKELETON PROGRAM uses the data structure `Answer`.

0	6	.	1
---	---	---	---

State what the second element of this data structure, `Answer[1]`, is used for. [1 mark]

0	6	.	2
---	---	---	---

Explain the purpose of the third element of this data structure, `Answer[2]`, AND how it is used. [3 marks]

0	7
---	---

The grid size of the puzzle is currently 9 and sub-grids are 3 x 3.

If the grid size were to be increased to 16, the sub-grids would be 4 x 4 and the hexadecimal digits 0 1 2 3 4 5 6 7 8 9 A B C D E F could be used to solve the puzzle.

Describe TWO changes that would need to be made to the subroutine

`DisplayGrid` to enable this.

You should not make any changes to the **SKELETON PROGRAM** to answer this question. [2 marks]

0	8
---	---

Explain what is meant by decomposition. [2 marks]

[Turn over]

0	9	.	1
---	---	---	---

Explain the differences between definite and indefinite iteration. [2 marks]

0	9	.	2
---	---	---	---

State the identifier of a subroutine in the SKELETON PROGRAM that contains DEFINITE iteration. [1 mark]

0	9	.	3
---	---	---	---

State the identifier of a subroutine in the SKELETON PROGRAM that contains INDEFINITE iteration. [1 mark]

1	0	.	1
---	---	---	---

Explain what exception handling is used for. [2 marks]

10.2

**State the identifier of a subroutine in the SKELETON PROGRAM that performs exception handling AND give an example of a circumstance that might cause an exception within that subroutine.
[2 marks]**

11.1

What is a subroutine? [1 mark]

11.2

The use of subroutines promotes code re-use.

Describe, with references to subroutine(s) in the SKELETON PROGRAM, an example of how a subroutine has been re-used. [1 mark]

[Turn over]

1 1 . 3

Describe an advantage of re-using subroutines. [1 mark]

1 1 . 4

FIGURE 4, on page 10 of the insert, shows an incomplete hierarchy chart for part of the SKELETON PROGRAM.

Complete TABLE 5, on the opposite page, by writing the labels that should appear in each of the boxes (a) to (d) in FIGURE 4.

TABLE 5

BOX	LABEL
(a)	
(b)	
(c)	
(d)	

Copy the contents of all the unshaded cells in TABLE 5 into your Electronic Answer Document. [2 marks]

[Turn over]

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	2
---	---

This question adds further validation to the **SKELETON PROGRAM**.

When a puzzle is loaded, some cells already contain numbers. These cells are referred to as protected cells.

FIGURE 5, on the opposite page, shows the numbers in the puzzle grid cells when `puzzle1` is first loaded. These are the protected cells for this puzzle.

FIGURE 5

	1	2	3	4	5	6	7	8	9
	==== .	==== .	====	==== .	==== .	====	==== .	==== .	====
1	8 .	. 5	 7	

2	9 .	. 4		5 .	. 4		

3	4 .	1 .		. 6	
	==== .	==== .	====	==== .	==== .	====	==== .	==== .	====
4		7 .	. .		1 .	6 .	

5	1 .	. .		4 .	. 6		. .	. 3	

6	. 5 .	. 8		. .	. 1		
	==== .	==== .	====	==== .	==== .	====	==== .	==== .	====
7 1 .			. 4 .	. 9	

8		2 .	. 7		. .	. 1	

9	2		5 .	. 6	
	==== .	==== .	====	==== .	==== .	====	==== .	==== .	====

[Turn over]

The SKELETON PROGRAM is to be changed so that the user cannot change the contents of any protected cells.

The subroutine `SolvePuzzle` needs to be modified so that the user cannot change a digit in a protected cell but can still enter a digit into an empty cell or change a digit that they have previously entered.

WHAT YOU NEED TO DO:

TASK 1

Amend the subroutine `SolvePuzzle` so that it checks every cell reference entered by the user against the protected cell references in `Puzzle` and only allows the contents of the cell to be changed if the cell referenced by `CellInfo` is not a protected cell.

If a protected cell is referenced, an appropriate error message should be displayed.

TASK 2

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

- **run your amended SKELETON PROGRAM**
- **enter P**
- **load puzzle1**
- **enter S**
- **enter 117**
- **enter 323**
- **enter 993**
- **enter 853**
- **enter 854**

[Turn over]

BLANK PAGE

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

1 2 . 1

Your **PROGRAM SOURCE CODE** for the entire subroutine `SolvePuzzle`.

[7 marks]

1 2 . 2

SCREEN CAPTURE(S) showing the requested test described in TASK 2.

The **SCREEN CAPTURE(S)** need(s) to show the puzzle grid before and after the described test. [1 mark]

[Turn over]

1	3
---	---

This question adds further validation to the **SKELETON PROGRAM**. The subroutine `SolvePuzzle` asks the user to enter coordinates and a digit. The **SKELETON PROGRAM** is to be changed so that if the digit entered by the user already exists in the referenced row, column or sub-grid, the digit cannot be used.

FIGURE 6, on the opposite page, shows the numbers in the puzzle grid cells of `puzzle1` when first loaded.

FIGURE 6

	1	2	3	4	5	6	7	8	9
	=====	.=====	.=====	=====	.=====	.=====	=====	.=====	.=====
1	8	.	.	5	7

2	9	.	.	5	.	.	4	.	.

3	4	.	1	.	.	6	.	.	.
	=====	.=====	.=====	=====	.=====	.=====	=====	.=====	.=====
4	.	.	.	7	.	.	1	.	6

5	1	.	.	4	.	.	6	.	3

6	.	5	.	8	.	.	1	.	.
	=====	.=====	.=====	=====	.=====	.=====	=====	.=====	.=====
7	1	.	.	4	.

8	.	.	.	2	.	.	7	.	1

9	2	5	.	6
	=====	.=====	.=====	=====	.=====	.=====	=====	.=====	.=====

[Turn over]

WHAT YOU NEED TO DO:

TASK 1

Write a new subroutine, `DuplicateDigit`, which takes `PuzzleGrid`, `Row`, `Column` and `Digit` as parameters. The subroutine should return `True` if the digit entered by the user is already present in the row, column or sub-grid of the cell the user has entered. Otherwise, the subroutine should return `False`.

TASK 2

Amend the subroutine `SolvePuzzle` so that it uses `DuplicateDigit` to check the user input and only allows the digit to be placed into the puzzle grid if the digit is not already present in the row, column or sub-grid of the cell the user has entered. If the digit is already present, an appropriate message should be given to the user.

TASK 3

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

- **run your amended SKELETON PROGRAM**
- **enter L**
- **load puzzle1**
- **enter S**
- **enter 178**
- **enter 819**
- **enter 124**
- **enter 989**
- **enter 555**

[Turn over]

BLANK PAGE

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

1 3 . 1

Your **PROGRAM SOURCE CODE** for the entire subroutine `DuplicateDigit` and the entire subroutine `SolvePuzzle`.

[8 marks]

1 3 . 2

SCREEN CAPTURE(S) showing the requested test described in TASK 3.

The **SCREEN CAPTURE(S)** need(s) to show the puzzle grid before and after the described test. [1 mark]

[Turn over]

1	4
---	---

This question extends the functionality of the SKELETON PROGRAM.

The user is to be allowed to clear the contents of the most recently changed cells. To do this, they will enter a negative integer when prompted to enter a row, column and digit. The integer will represent the number of cells to be cleared.

For example, if the user enters -3 , the three most recently entered digits should each be replaced by a space in the puzzle grid and the `Answer` data structure should be updated.

If the number of cells to be cleared is greater than the number of entries in the `Answer` data structure, all digits entered by the user should be cleared.

WHAT YOU NEED TO DO:**TASK 1**

Write a new subroutine, `ClearEntries`, that clears the required number of previously entered digit(s) from the puzzle grid and updates the `Answer` data structure as described above.

TASK 2

Amend the subroutine `SolvePuzzle` to test user input for a negative value. If the user input is a negative integer, the subroutine should call `ClearEntries` with the necessary parameters and then re-display the puzzle grid.

[Turn over]

TASK 3

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

- **run your amended SKELETON PROGRAM**
- **enter P**
- **load puzzle1**
- **enter S**
- **enter -x**
- **enter -1**
- **enter -5**

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

14.1

Your **PROGRAM SOURCE CODE** for the entire subroutine `ClearEntries` and the entire subroutine `SolvePuzzle`.

[12 marks]

14.2

SCREEN CAPTURE(S) showing the requested test described in TASK 3.

The **SCREEN CAPTURE(S)** need(s) to show the puzzle grid before and after the described test. [1 mark]

END OF QUESTIONS

BLANK 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 © 2022 AQA and its licensors. All rights reserved.

IB/M/CD/Jun22/7516/1/E2



2 2 6 A 7 5 1 6 / 1