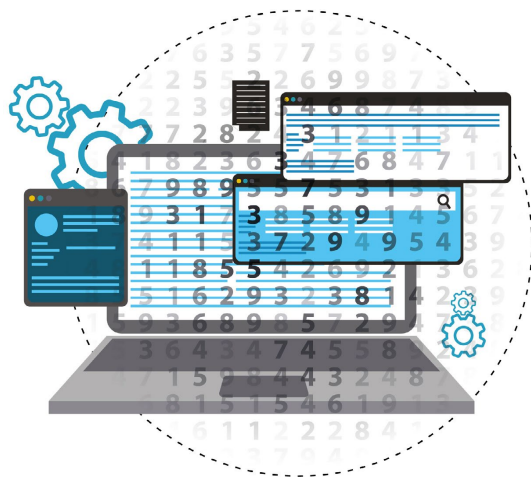


Example Responses – Paper 2

Cambridge International AS & A Level Computer Science 9618

For examination from 2024



Contents

Introduction.....	4
Question 1	5
Question 2	8
Question 3	12
Question 4	14
Question 5	16
Question 6	18
Question 7	22
Question 8	25

Introduction

The main aim of this booklet is to exemplify standards for those teaching Cambridge International AS & A Level Computer Science.

This booklet contains responses to all questions from June 2024 Paper 22, which have been written by a Cambridge examiner. Responses are accompanied by a brief commentary highlighting common errors and misconceptions where they are relevant.

The question papers and mark schemes are available to download from the [School Support Hub](#)

9618 June 2024 Question Paper 22

9618 June 2024 Mark Scheme 22

9618 June 2024 Mark Scheme 22

Past exam resources and other teaching and learning resources are available from the [School Support Hub](#)

Question 1

- 1 (a) The following table contains pseudocode examples.

Each example may contain statements that relate to one or more of the following:

- selection
- iteration (repetition)
- input/output.

Complete the table by placing **one or more** ticks (✓) in each row.

Pseudocode example	Selection	Iteration	Input/Output
FOR Index ← 1 TO 10 Data[Index] ← 0 NEXT Index		✓	
WRITEFILE ThisFile, "*****"			✓
UNTIL Level > 25		✓	
IF Mark > 74 THEN READFILE OldFile, Data ENDIF	✓		✓

[4]

Examiner comment

Most candidates were awarded at least 1 or 2 marks. The directive to place 'one or more ticks' seemed to have been generally well understood.

- (b) Program variables have data types as follows:

Variable	Data type
MyChar	CHAR
MyString	STRING
MyInt	INTEGER

Complete the table by filling in each gap with a function (from the **insert**) so that each expression is valid.

Expression
MyInt ← <i>INT</i> (3.1415926)
MyChar ← <i>MID</i> ("Elwood", 3, 1)
MyString ← <i>NUM_TO_STR</i> (..... <i>INT</i> (27.509))
MyInt ← <i>STR_TO_NUM</i> (..... <i>RIGHT</i> ("ABC123", 3))

[4]

Examiner comment

- Many candidates were awarded full marks.
- A few candidates wrote the full function header (from the insert) rather than just the function name.
- Common errors included:
 - choosing a function that didn't deliver an object of the correct type for assignment to the variable on the left-hand side
 - incorrectly naming functions from the insert e.g. NUM_TO_STRING().

- (c) The variables given in part (b) are chosen during the design stage of the program development life cycle.

The choices are to be documented to simplify program maintenance.

State a suitable way of documenting the variables **and** give **one** piece of information that should be recorded, in addition to the data type.

Variables can be documented using an identifier table. The use of the variable should be recorded.

[2]

Examiner comment

- Many candidates seemed to have missed the keyword 'documenting' in the question and suggested that variables should be stored in an array record or file.
- Some candidates suggested the variable could be 'documented in an array' and 'use comments' was seen frequently.
- Candidates needed to read each question carefully to make sure they understood what was required before forming their answer.

Question 2

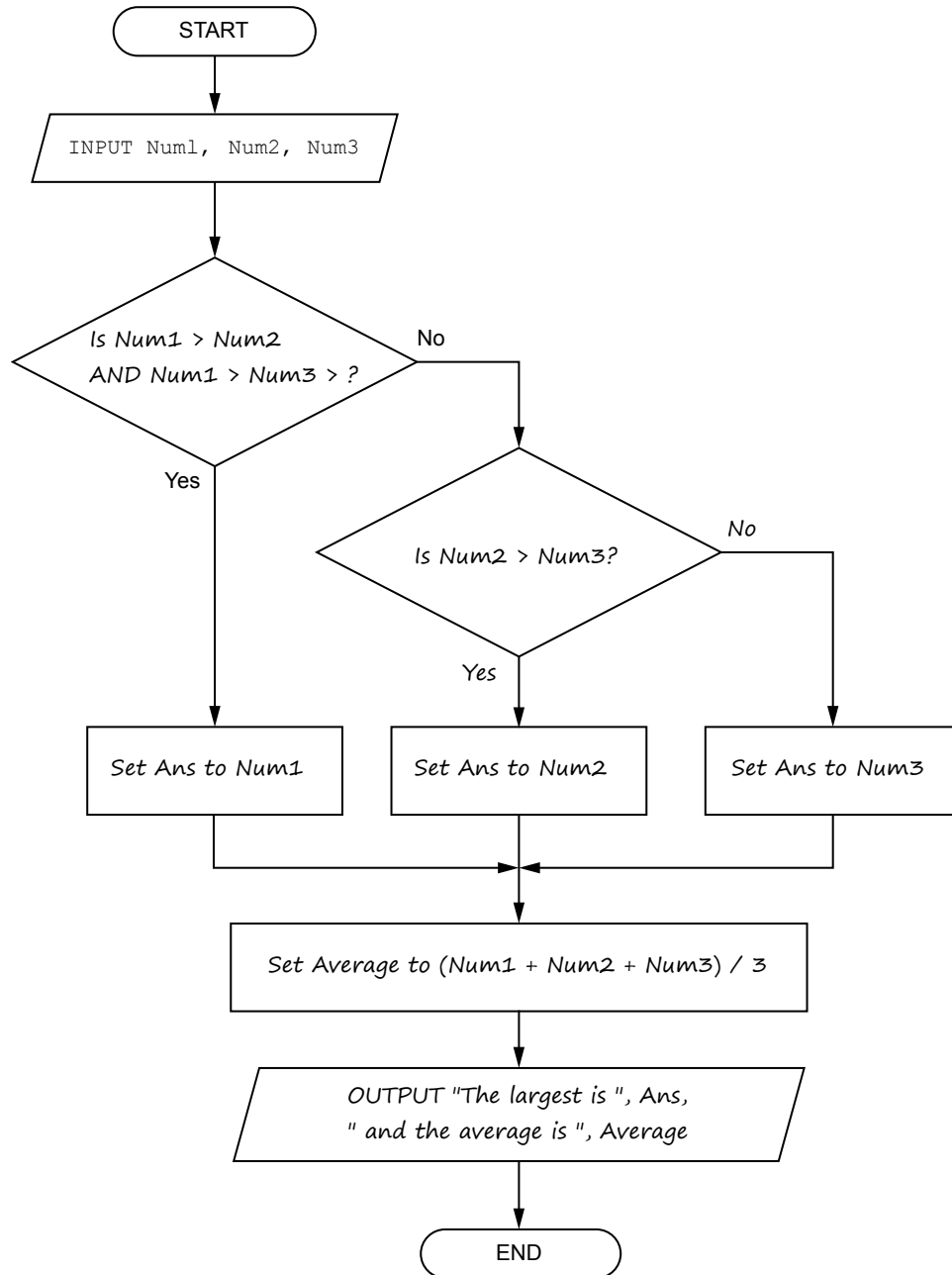
2 A program is being developed.

(a) An algorithm for part of the program will:

- input three numeric values and assign them to identifiers Num1, Num2 and Num3
- assign the largest value to variable Ans
- output a message giving the largest value and the average of the three numeric values.

Assume the values are all different and are input in no particular order.

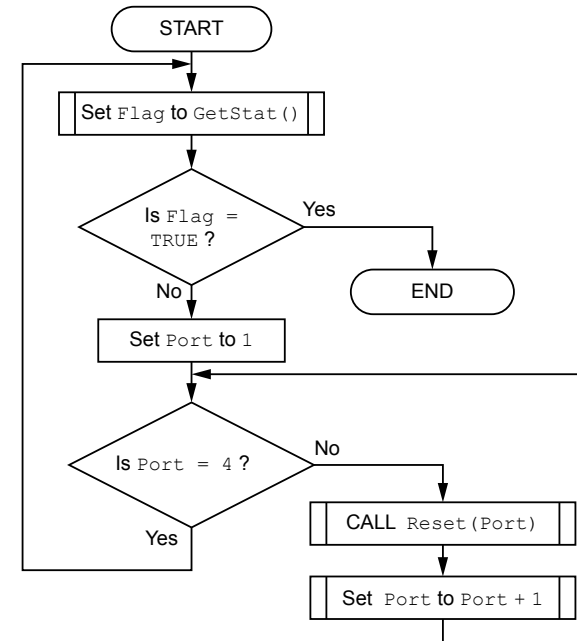
Complete the program flowchart on page 5 to represent the algorithm.



Examiner comment

- A small number of perfect solutions were awarded full marks.
- Common errors included:
 - using incorrect test syntax in the first decision diamond, for example: `Num1 > Num2 AND Num3`
 - reversing the assignment: `Set Num1 TO Ans`
 - omitting the labels on arrows from the second decision diamond
 - making an incorrect average calculation, often omitting brackets: `Num1 + Num2 + Num3 / 3`
 - a lack of separators in the OUTPUT statement and sometimes omission of the keyword itself.

(b) A different part of the program contains an algorithm represented by the following program flowchart:



Write pseudocode for the algorithm.

Write pseudocode for the algorithm.

```

Flag ← GetStat()
WHILE Flag <> TRUE
    Port ← 1
    REPEAT
        CALL Reset(Port)
        Port ← Port + 1
    UNTIL Port = 4
    Flag ← GetStat()
ENDWHILE

```

[5]

Examiner comment

- The quality of responses suggested that this type of task was not something candidates had much experience of.
- Common errors included:
 - using unterminated clauses
 - using interleaved rather than nested clauses
 - using programmatic jumps in the form of `BREAK`
 - using (one or more) `ENDPROCEDURE` statements within the algorithm
 - incorrect use of the `GetStat()` function, such as:
 - `CALL GetStat (Flag)`
 - `GetStat () → Flag.`

Question 3

- 3 A factory needs a program to help manage its production of items.

Data will be stored about each item.

The data for each item will be held in a record structure of type `Component`.

The programmer has started to define the fields that will be needed as shown in the table.

Field	Example value	Comment
Item_Num	123478	a numeric value used as an array index
Reject	FALSE	TRUE if this item has been rejected
Stage	'B'	a letter to indicate the stage of production
Limit_1	13.5	any value in the range 0 to 100 inclusive
Limit_2	26.4	any value in the range 0 to 100 inclusive

- (a) (i) Write pseudocode to declare the record structure for type `Component`.

```

TYPE Component
    DECLARE Item_Num : INTEGER
    DECLARE Reject : BOOLEAN
    DECLARE Stage : CHAR
    DECLARE Limit_1 : REAL
    DECLARE Limit_2 : REAL
ENDTYPE

```

[4]

Examiner comment

- This question was well answered by most candidates.
- Common mistakes included:
 - omitting the keyword `DECLARE`
 - omitting `ENDTYPE`
 - adding `DECLARE` before the `TYPE` header
 - using dot notation for the individual fields within the record
 - specifying a range of values for the two `Limit` fields.

- (ii) A 1D array `Item` of 2000 elements will store the data for all items.

Write pseudocode to declare the `Item` array.

`DECLARE Item : ARRAY [1:2000] OF Component`
 [2]

Examiner comment

- Many candidates answered this question well.
- Common mistakes included:
 - a missing or incorrect array type (commonly specified as `STRING`)
 - using a comma rather than a colon, suggesting a 2D array
 - omitting the keyword `DECLARE`.

- (b) State **three** benefits of using an array of records to store the data for all items.

- A loop can be used to search for a particular record.*
 [3]
- The array index can be used to directly access a field in a record e.g.*
Item[27].Reject

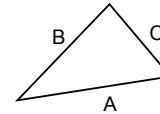
- The program is simpler and easier to write.*

Examiner comment

Some candidates just referred to the use of an array, rather than specifically to an array of records as the question asked. Answers such as 'the data is easier to access' were not awarded marks.

Question 4

- 4 A triangle has sides of length A, B and C.



In this example, A is the length of the longest side.

This triangle is said to be right-angled if the following equation is true:

$$A \times A = (B \times B) + (C \times C)$$

A procedure will be written to check whether three lengths represent a right-angled triangle. The lengths will be input in any sequence.

The procedure `IsRA()` will:

- prompt and input three integer values representing the three lengths
- test whether the three lengths correspond to the sides of a right-angled triangle
- output a suitable message.

The length of the longest side may **not** be the first value input.

Write pseudocode for the procedure `IsRA()`.

```

PROCEDURE IsRA()
  .....
  DECLARE a, b, c : INTEGER
  .....
  DECLARE RightAngled : BOOLEAN
  .....
  RightAngled ← FALSE
  .....
  OUTPUT "Input the lengths of the three sides"
  INPUT a, b, c
  .....
  IF a*a = (b*b) + (c*c) THEN
    RightAngled ← TRUE
  ELSIF b*b = (a*a) + (c*c) THEN
    RightAngled ← TRUE
  .....

```

```

ELSEIF  $c^2 = (a^2) + (b^2)$  THEN
    RightAngled  $\leftarrow$  TRUE
ENDIF

IF RightAngled = TRUE THEN
    OUTPUT "It is right-angled"
ELSE
    OUTPUT "Not right-angled"
ENDIF

ENDPROCEDURE

```

Examiner comment

- Most solutions included the basic 'building' blocks of the algorithm as follows:
 - Prompt and input the three values.
 - Attempt to determine the longest side / combine three tests.
 - Perform the test(s).
 - Output one of two messages.
- Common errors included:
 - passing the three lengths as parameters to the procedure
 - omitting the prompt before the input
 - using the alphabetic character 'x' rather than the symbol "*" as the multiplication operator
 - omitting the keyword `ENDIF` from the clause which selects between the two possible output statements.
- A common mistake was the attempted use of the logical operator `AND` as some form of continuation, in statements such as:


```
Longest ← Num1 AND Side1 ← Num2 AND Side2 ← Num3
```
- Some solutions used an array for the three input values. This was not considered to be necessary for just three values and in many cases the increased complication introduced errors.
- Care is taken to provide the number of answer lines required by a 'typical' solution and if candidates find that their solution is taking many more lines (or even pages) then they probably need to rethink their answer.

Question 5

- 5** A program is being designed in pseudocode.

The program contains a global 1D array `Data` of type `string` containing 200 elements.

The first element has the index value 1.

A procedure `Process()` is written to initialise the values in the array:

```

PROCEDURE Process(Label : STRING)
  DECLARE Index : INTEGER
  Index ← 0
  INPUT Data[Index]
  WHILE Index < 200
    Index ← Index + 1
    CASE OF (Index MOD 2)
      0 : Data[Index] ← TO_UPPER(Label)
      1 : Data[Index] ← TO_LOWER(Label)
    OTHERWISE : OUTPUT "Alarm 1201"
  ENDCASE
  NEXT Index
  OUTPUT "Completed " & Index & " times"
ENDPROCEDURE

```

- (a) (i) The pseudocode contains **two** syntax errors and **one** other error.

Identify the errors.

Syntax error 1 ENDWHILE is missing and NEXT is used instead to end the loop

Syntax error 2 cannot use & operator with an integer

Other error `data[0]` is accessed but array starts at index 1

[3]

Examiner comment

- Although many candidates identified all three errors, often they were placed in the wrong category and so the corresponding mark was not awarded.
- A few candidates provided a corrected line of pseudocode, contrary to the wording of the question.

- (ii) The procedure contains a statement that is **not** needed.

Identify the pseudocode statement **and** explain why it is **not** needed.

Statement *the OTHERWISE statement*

Explanation *any number MOD 2 can only be 0 or 1 and both these values are already checked*

[2]

Examiner comment

- Answers that correctly identified the `OTHERWISE` statement for the first mark often did not provide an adequate explanation.
- Many candidates felt that the final output statement was not needed and this was awarded the mark, provided the explanation referred to the loop being for a known number of iterations.
- There were many No-Responses (NR) for this question.

- (b) After correcting all syntax errors, the pseudocode is translated into program code which compiles without generating any errors.

When the program is executed it unexpectedly stops responding.

Identify the type of error that has occurred.

run-time

[1]

Examiner comment

This question was correctly answered by most candidates. Incorrect answers included a range of terms from the syllabus, supplemented with possible error types such as 'human', 'machine', 'value', 'interrupt' and 'system'.

Question 6

- (a) Up to 16 characters can be displayed. Track titles longer than 16 characters will need to be trimmed as follows:
- Words must be removed from the end of the track title until the resulting title is less than 14 characters.
 - When a word is removed, the space in front of that word is also removed.
 - Three dots are added to the end of the last word displayed when one or more words have been removed.

The table below shows some examples:

Original title	Display string															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bat out of Hull	B	a	t		o	u	t		o	f		H	u	l	l	
Bohemian Symphony	B	o	h	e	m	i	a	n	.	.	.					
Paperbook Writer	P	a	p	e	r	b	o	o	k		W	r	i	t	e	r
Chris Sings the Blues	C	h	r	i	s		S	i	n	g	s	.	.	.		
Green Home Alabama	G	r	e	e	n		H	o	m	e	.	.	.			

A function `Trim()` will:

- take a string representing the original title
- return the string to be displayed.

Assume:

- Words in the original title are separated by a single space character.
- There are no spaces before the first word or after the last word of the original title.
- The first word of the original title is less than 14 characters.

Write pseudocode for the function `Trim()`.

```

FUNCTION Trim(Title : STRING) RETURNS STRING
    CONSTANT Space = ' '

    IF LENGTH(Title) < 17 THEN
        RETURN Title
    ENDIF

    WHILE LENGTH(Title) > 13
        REPEAT

```

Write pseudocode for the function `Trim()`.

```

Title ← LEFT(Title, LENGTH(Title) - 1)
UNTIL RIGHT(Title, 1) = Space
ENDWHILE

Title ← LEFT(Title, LENGTH(Title) - 1) & "..."
RETURN Title

ENDFUNCTION

```

[7]

Examiner comment

- This pseudocode question provided a wide range of answers, from virtually no attempt to fully working solutions.
- Most solutions included at least one loop. The requirement for a conditional loop prevented many count-controlled solutions being awarded this mark as they lacked an immediate `RETURN` or `BREAK`.
- MP1: Many candidates were awarded this mark, although the absence of `ENDFUNCTION` prevented this being awarded on several occasions.
- MP2: Many candidates attempted this, but their responses often consisted of just the initial length test and lacked the corresponding `RETURN`. In many cases also the corresponding `ENDIF`. When these were present, a common mistake was to return an uninitialized variable (e.g. `NewLine`) or one that had been initialised to an empty string rather than the original parameter.
- MP3: This was a design mark requiring only a conditional construct and was awarded in many cases.
- MP4: This was included less frequently than the MP2 test and often the length test was 'one out', so for example the solution would stop at index position 14 and then candidates would attempt to concatenate three dots.
- MP5: The use of an inner loop was not common. This mark was often awarded for testing whether an extracted character was a space character.
- MP6: This was another design mark which was awarded for any competent use of the `LEFT()` function and in reasonable attempts this often followed MP5.
- MP7: This mark was rarely awarded as it effectively required a completely working algorithm (albeit without MP8).
- MP8: Many candidates were awarded MP8, often as a Follow-Through or Benefit-of-Doubt mark.
- Common errors included:
 - a missing end-of-clause keyword
 - using the keyword `LENGTH` as an identifier
 - using '+' to concatenate
 - treating a string as an array.
- Some candidates gave a solution based on a loop that worked back from string index position 14 to the first space character and this was often awarded full marks.

(b) Music is stored as a sequence of digital samples.

Each digital sample is a denary value in the range 0 to 99999999 (8 digits).

The samples are to be stored in a text file. Each sample is converted to a numeric string and 32 samples are concatenated (joined) to form a single line of the text file.

Each numeric string is 8 characters in length; leading '0' characters are added as required.

Example:

Sample	Denary value	String
1	456	"00000456"
2	48	"00000048"
3	37652	"00037652"
		
32	673	"00000673"

The example samples will be stored in the text file as a single line:

"000004560000004800037652...00000673"

(i) Identify one drawback of adding leading '0' characters to each numeric string.

Leading '0' characters make the file very big.

[1]

Examiner comment

- Many answers were too vague. Phrases such as 'makes the string bigger' or 'uses more space' were not awarded the mark. Candidates needed to use appropriate and precise technical language wherever possible.
- A few candidates referred to the spaces being 'not needed' which mapped to the MS 'redundant'.

(ii) Suggest an alternative method of storing the samples which does **not** involve adding leading '0' characters but which would still allow each individual sample to be extracted.

Replace the '0' characters with a non-numeric separator character, such as a comma.

[1]

Examiner comment

Some answers were considered too vague, such as 'replace the zeroes with spaces'. Candidates needed to use clear and precise wording.

- (iii) State **one** drawback of the alternative method given in part (b)(ii).

The program to recover the original samples is more complex.
 [1]

Examiner comment

Some candidates gave very vague answers which were not relevant. Others got close to identifying the drawback, but some of those did not reference the terms in bold on the Mark Scheme (or their equivalent) so this mark was often not awarded.

Question 7

- 7 A fitness club has a computerised membership system.

The system stores information for each club member: name, home address, email address, mobile phone number, date of birth and exercise preferences.

Many classes are full, and the club creates a waiting list for each class. The club adds details of members who want to join a class that is full to the waiting list for that class.

When the system identifies that a space is available in one of the classes, a new module will send a text message to each member who is on the waiting list.

- (a) Decomposition will be used to break the new module into sub-modules (sub-problems).

Identify **three** sub-modules that could be used in the design **and** describe their use.

Sub-module 1 *get details*.....

Use *get the phone number and name of a member who is on the waiting list*

Sub-module 2 *write text*.....

Use *write a message to send to a club member*

Sub-module 3 *send text*.....

Use *send a text message to the club member telling them that a space is available*

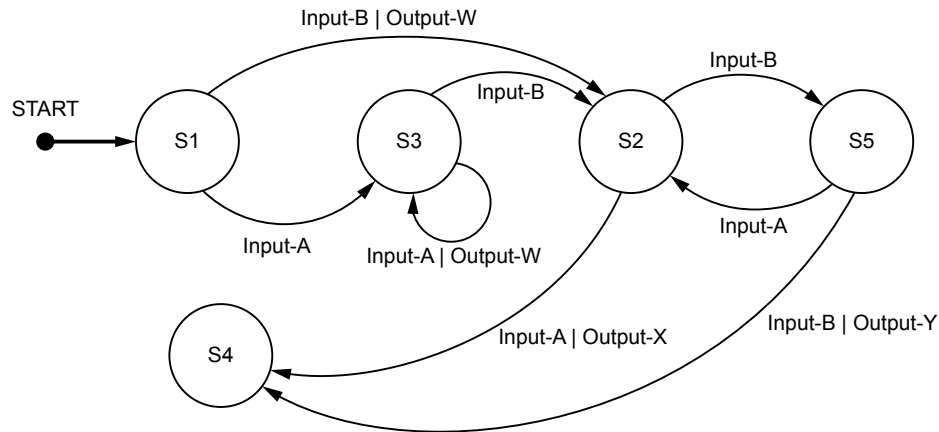
[3]

Examiner comment

- There were only a few completely correct answers.
- Most answers did not focus on the requirements of the new module as the question asked and instead suggested various modules that might be of use in the complete membership system.

- A common mistake was to suggest modules that setup initial club member details or which added members to the waiting list.
- Several solutions referred to the sending of an email rather than a text. This error suggests that candidates had not read the question carefully enough.
- A small number of candidates incorrectly suggested computing terms such as module types or design methodologies.

(b) A different part of the program is represented by the following state-transition diagram.



(i) Complete the table to show the inputs, outputs and next states.

Assume that the current state for each row is given by the 'Next state' on the previous row. For example, the first Input-A is made when in state S1.

If there is no output for a given transition, then the output cell should contain 'none'.

The first two rows have been completed.

Input	Output	Next state
		S1
Input-A	none	S3
Input-A	Output-W	S3
Input-B	none	S2
Input-B	none	S5
Input-A	none	S2
Input-A	Output-X	S4

[5]

Examiner comment

Many candidates answered this question well, and most answers were awarded full marks.

(ii) Identify the input sequence that will cause the minimum number of state changes in the transition from S1 to S4.

input-B followed by input-A [1]

Examiner comment

This was a question that possibly approached the topic in a slightly unexpected way which resulted in many incorrect answers often listing the states themselves or the number of state changes (in both cases often incorrectly).

Question 8

- 8 A teacher is designing a program to process pseudocode projects written by her students.

Each student project is stored in a text file.

The process is split into a number of stages. Each stage performs a different task and creates a new file.

For example:

File name	Comment
MichaelAday_src.txt	Student project file produced by student Michael Aday
MichaelAday_S1.txt	File produced by stage 1
MichaelAday_S2.txt	File produced by stage 2

- (a) Suggest a reason why the teacher's program has been split into a number of stages **and** give the benefit of producing a different file from each stage.

Reason *It makes the program easier to debug.*

Benefit *The operation of each stage can be checked separately.*

[2]

Examiner comment

- Many candidates focused on the teaching activity (for example, 'the teacher can easily see the errors made by her students') rather than the advantages as applied to the program.
- Where marks were awarded, this was usually for a 'reason' from the list given in the Mark Scheme.
- Candidates rarely referred to 'each stage'.

- (b) The teacher has defined the first program module as follows:

Module	Description
DeleteSpaces()	- called with a parameter of type string representing a line of pseudocode from a student's project file - returns the line after removing any leading space characters The following example shows a string before and after the leading spaces have been removed: Before: " IF X2 > 13 THEN" After: "IF X2 > 13 THEN"

Complete the pseudocode for module DeleteSpaces().

FUNCTION DeleteSpaces(Line : STRING) RETURNS STRING

.....*DECLARE NewLine : STRING*.....

.....*DECLARE Index, SpaceCount : INTEGER*.....

.....*DECLARE NextChar : CHAR*.....

.....*CONSTANT Space = ' '*.....

.....*SpaceCount ← 0*.....

.....*FOR Index ← 1 TO LENGTH(Line)*.....

.....*NextChar ← MID(Line, Index, 1)*.....

.....*IF NextChar <> Space AND SpaceCount = 0 THEN*.....

.....*SpaceCount ← Index*.....

.....*ENDIF*.....

.....*NEXT Index*.....

.....*IF SpaceCount = 0 THEN //Line is all spaces*.....

.....*NewLine ← "" //so delete them all*.....

.....*ELSE*.....

.....*NewLine ← RIGHT(Line, LENGTH(Line) - SpaceCount - 1)*.....

.....*ENDIF*.....

.....*RETURN NewLine*.....

ENDFUNCTION

[6]

Examiner comment

- Although not relevant to the question, many solutions included file handling operations. Often these solutions included a statement which read a value for `Line` from a file, and so prevented a mark being awarded (usually MP2) due to `Line` then containing an incorrect value.
- MP1: the count-controlled loop to the length of `Line` was seen frequently. A common mistake was to omit the assignment arrow, i.e. `FOR index 1 to LENGTH(Line)`. A function `LEN()` was seen several times. This is not a function from the insert.
- MP2: together with MP1, MP2 was most usually awarded. Both `MID()` and `LEFT()` were often used correctly.
- MP3: many count-controlled solutions simply counted every space in `Line` so were not awarded MP3.
- MP4: This was a design mark awarded for the correct use of `RIGHT()` or `MID()` functions. Solutions that attempted to build a new string character by character were also awarded this mark.
- MP5: few solutions were awarded this mark as it required the completely correct generation of the new string. Where sub-string functions were used, it was common for the length parameter to be out by one.
- MP6: this was often awarded as a Follow-Through or Benefit-of-Doubt mark. As is usual, a number of solutions incorrectly used `OUTPUT` rather than `RETURN`.
- A few effective solutions were based around a simple `WHILE` loop which repeatedly reduced `Line` until the first character was a non-space, as below:

```
WHILE Line <> "" AND LEFT(Line, 1) = Space
    Line ← RIGHT(Line, LENGTH(Line) - 1)
ENDWHILE

RETURN Line
```

(c) Two modules are defined:

Module	Description
<code>DeleteComment()</code> (already written)	- called with a parameter of type string representing a line of pseudocode from a student's project file - returns the line after removing any comment
<code>Stage_2()</code>	- called with two parameters: - a string representing an input file name - a string representing an output file name - copies each line from the input file to the existing output file having first removed all leading spaces and comments from that line - does not write blank lines to the output file - outputs a final message giving the number of blank lines removed

Write pseudocode for module `Stage_2()`.

Modules `DeleteComment()` and `DeleteSpaces()` must be used in your solution.

```
PROCEDURE Stage_2(InFile, OutFile : STRING)
.....

DECLARE Line : STRING
.....

DECLARE BlankCount: INTEGER
.....

BlankCount ← 0
.....

OPEN InFile FOR READ
.....

OPEN OutFile FOR APPEND
.....

WHILE NOT EOF(InFile)
.....

    READFILE InFile, Line
.....

    Line ← DeleteSpaces(Line)
.....

    Line ← DeleteComment(Line)
.....

    IF Line <> "" THEN
.....

        WRITEFILE OutFile, Line
.....

    ELSE
.....

        BlankCount ← BlankCount + 1
.....

    ENDIF
.....

ENDWHILE
.....

CLOSEFILE InFile
.....

CLOSEFILE OutFile
.....

OUTPUT "Number of lines removed was ", BlankCount
.....

ENDPROCEDURE
```

[8]

Examiner comment

- Answers generally seemed to be awarded higher marks for this question than for the previous one.
- Many very good solutions were seen, but there were also several with no attempt made.
- The filename presented problems to many. Often the parameter identifier incorrectly had a `'.txt'` suffix added and it was very common for the identifier to be enclosed in double quotation marks when subsequently used.
- MP1: this was awarded to most solutions and in some cases was the only mark awarded. Some answers consisted of just two lines: the procedure heading and ending suggesting that candidates have learnt from previous mark schemes. Two occasional errors included:
 - suggesting the module was a `FUNCTION`
 - using the keywords `INPUT` and `OUTPUT` as file identifiers.
- MP2: this was rarely awarded. The output file was usually opened in `WRITE` mode. `CLOSEFILE` statements were regularly omitted or lacked a filename.
- MP3: this was one of the marks available which together with MP1 and MP4 gave a 3-mark base for most solutions.
- Common mistakes included:
 - omitting the parameter from `EOF()`
 - missing `ENDWHILE`
 - use of a count-controlled loop: `FOR Index ← 1 to EOF(InputFile)`
 - use of a python-like syntax: `FOR Line IN InputFile.`
- MP4: this was one of the more accessible marks. One mistake seen occasionally was the inclusion of a line index, often as a third parameter. The scope of Paper 2 does not extend beyond simple text files which are accessed serially.
- MP5: this was not well answered. Common mistakes included the use of keyword `CALL` and not using the value returned by the first function as the parameter to the second.
- MP6: even given the problems caused by not correctly using the functions in MP5, many solutions did not test the return string before performing the `WRITEFILE`. The usual approach was to test for an empty string immediately following the `READFILE` operation rather than after the two 'delete' functions.
- MP7: this was often awarded for testing the original line read from file. It was noted that more solutions are now correctly initialising variables before using them to count events.
- MP8: this was often awarded for outputting the value of the variable used for counting, and frequently awarded as a Follow-Through mark if this variable had not been initialised. In some cases where the output also included a text message, the attempt to concatenate a string with an integer value prevented this mark being awarded.