



## Cambridge International A Level

CANDIDATE  
NAME

--

CENTRE  
NUMBER

--	--	--	--	--

CANDIDATE  
NUMBER

--	--	--	--



**COMPUTER SCIENCE**

**9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills

**May/June 2021**

**2 hours**

You must answer on the question paper.

No additional materials are needed.

### INSTRUCTIONS

- Answer **all** questions.
- Use a black or dark blue pen.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- You may use an HB pencil for any diagrams, graphs or rough working.
- Calculators must **not** be used in this paper.

### INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.

This document has **24** pages. Any blank pages are indicated.

1 (a) (i) State how characters are represented using the ASCII character set.

.....  
 .....  
 .....  
 ..... [2]

(ii) String values may be represented by a sequence of ASCII characters.

The following table shows consecutive memory locations.

Complete the table by adding the values to show how the string "FADED" may be stored in memory using the ASCII character set.

Refer to the **Appendix** on pages 22–23 for the list of built-in pseudocode functions and operators, which includes a reference to the `ASC ()` function.

Memory location	ASCII character value
100	
101	
102	
103	
104	

[2]

(b) Individual elements in a 1D array are referenced using an integer value.

In the pseudocode expression `StockID[n]`, the integer value is represented by the variable `n`.

(i) Give the technical terms for the minimum and maximum values for the variable `n`.

Minimum value .....

Maximum value ..... [1]

(ii) Give the correct term for the variable `n` in the pseudocode expression `StockID[n]`.

..... [1]

- (c) Each pseudocode statement in the following table may contain an error due to the incorrect use of the function or operator.

Describe the error in each case, **or** write 'NO ERROR' if the statement contains no error.

Refer to the **Appendix** on pages 22–23 for the list of built-in pseudocode functions and operators.

Statement	Error
Code ← LEFT("Cat", 4)	
Status ← MID("Aardvark", 0, 5)	
Size ← LENGTH("Password)	
Stock[n] ← Stock[n+1]	
Result ← 3 OR 4	

[5]

- 2 (a) The following pseudocode algorithm counts the number of each alphabetic character in the string `Msg`. The character count values are stored in an array `CharCount`.

Variable declarations are not shown.

```
FOR Index ← 1 TO 26
```

```
    CharCount[Index] ← 0
```

```
ENDFOR
```

```
FOR Index ← 1 TO LENGTH(Msg)
```

```
    ThisChar ← MID(Msg, Index, 1)
```

```
    ThisChar ← LCASE(ThisChar)
```

```
    IF ThisChar >= 'a' AND ThisChar <= 'z'
```

```
        THEN
```

```
            ThisIndex ← ASC(ThisChar) - 96 // value from 1 to 26
```

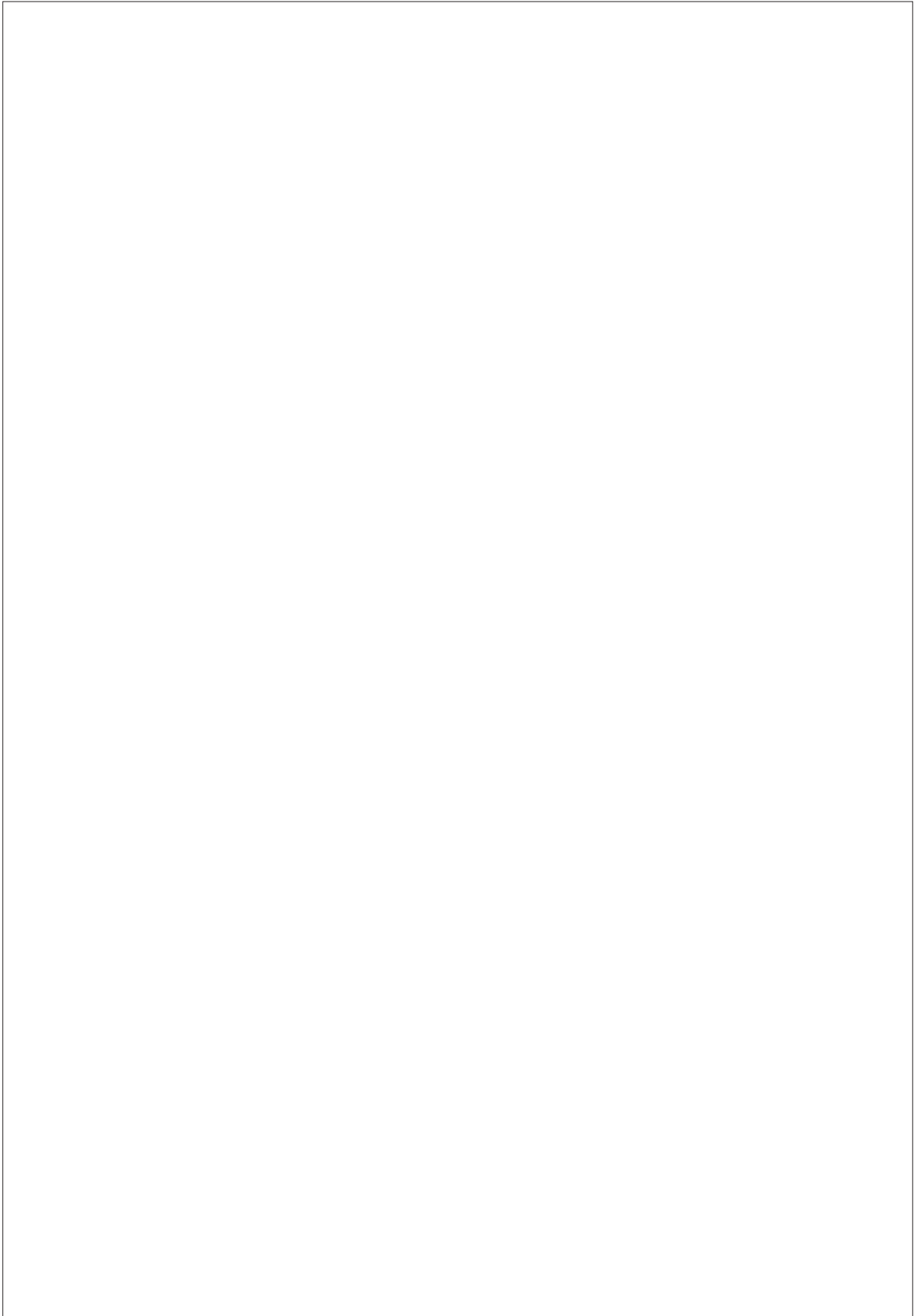
```
            CharCount[ThisIndex] ← CharCount[ThisIndex] + 1
```

```
        ENDIF
```

```
ENDFOR
```

Draw a program flowchart to represent the algorithm.

Variable declarations are not required in program flowcharts.



[5]

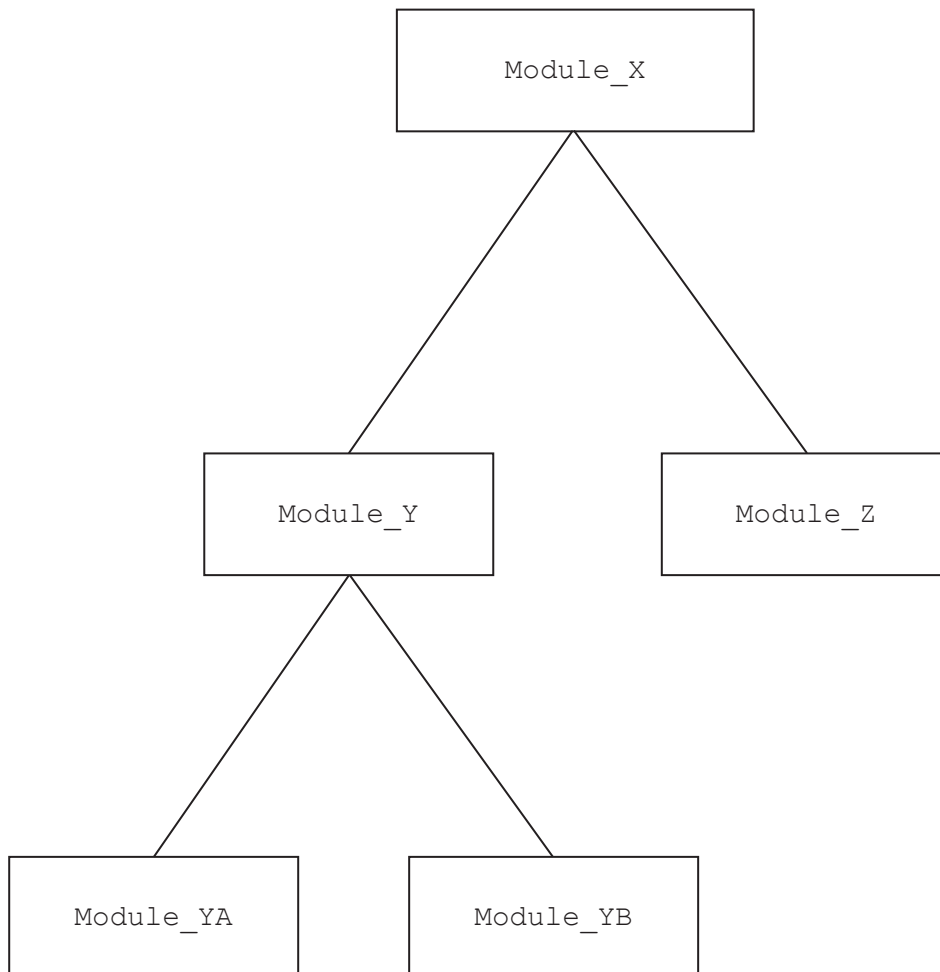




- 3 (a) The following table contains information about five modules in a program. It describes the calls made and the parameters passed.

Module	Description
Module_X	<ul style="list-style-type: none"> <li>repeatedly calls Module_Y then Module_Z</li> <li>passes a parameter of type REAL to Module_Y</li> <li>passes two parameters of type INTEGER to Module_Z</li> </ul>
Module_Y	calls either Module_YA or Module_YB
Module_Z	called with two parameters of type INTEGER
Module_YA	<ul style="list-style-type: none"> <li>called with a parameter of type REAL</li> <li>parameter is passed by reference</li> </ul>
Module_YB	<ul style="list-style-type: none"> <li>called with a parameter of type INTEGER</li> <li>returns a BOOLEAN value</li> </ul>

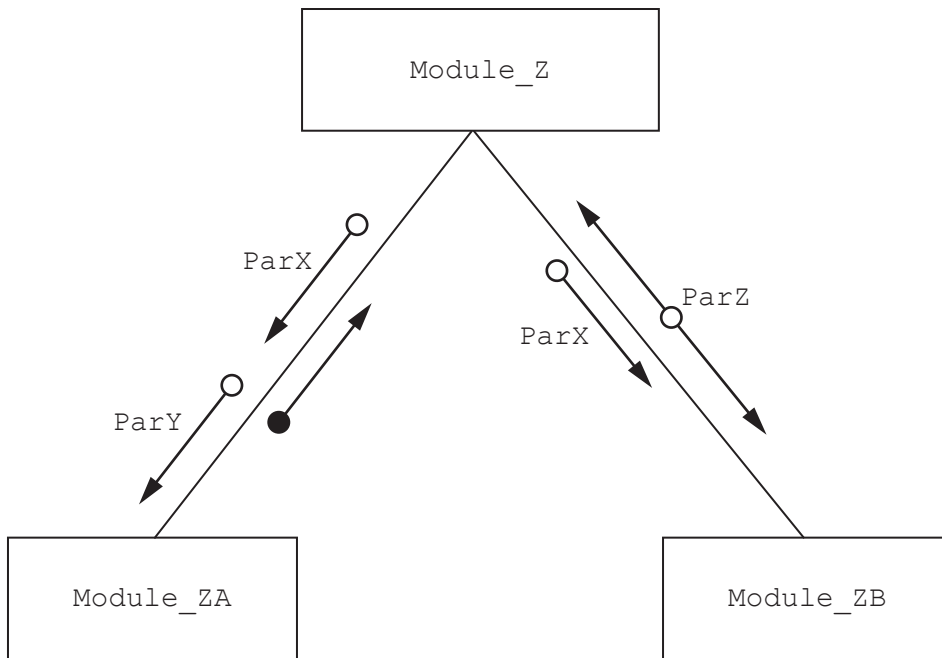
Complete the structure chart to include the information given about the five modules.



[5]



(b) Two more modules are added to the chart below `Module_Z` as shown:



Parameter data types are:

ParX : REAL  
 ParY : INTEGER  
 ParZ : STRING

(i) State whether `Module_ZA()` is a function or a procedure **and** justify your choice.

.....  
 .....  
 ..... [2]

(ii) Write the **pseudocode** header for `Module_ZB()`.

.....  
 .....  
 ..... [3]

4 The following is part of a program written in pseudocode:

```
DECLARE ThisArray : ARRAY[1:1000] OF STRING
DECLARE ArrayResult : INTEGER
```



```
PROCEDURE ScanArray(SearchString : STRING)

  DECLARE Index, Total : INTEGER
  DECLARE Error : BOOLEAN

  Index ← 1
  Total ← 0
  Error ← FALSE

  WHILE Index <= 1000 AND Error <> TRUE
    IF LENGTH(ThisArray[Index]) > 5
      THEN
        IF ThisArray[Index] = SearchString
          THEN
            Total ← Total + LENGTH(ThisArray[Index])
          ENDIF
        Index ← Index + 1
      ELSE
        Error ← TRUE
      ENDIF
    ENDWHILE

  ArrayResult ← INT(Total / (Index - 1))

ENDPROCEDURE
```

The procedure `ScanArray()` is amended as follows:

- `SearchString` is compared with just the first **four** characters of each array element.
- The total ignores the first **five** characters of each array element.
- When calculating `ArrayResult`, prevent any possible division by zero.

Refer to the **Appendix** on pages 22–23 for the list of built-in pseudocode functions and operators.



(b) Context-sensitive prompts are a feature of a typical Integrated Development Environment (IDE).

Explain the term **context-sensitive prompt**.

.....  
.....  
.....  
..... [2]

(c) (i) Identify the first stage in the program development cycle.

State the tasks that are completed during this stage.

First stage .....

Tasks .....

.....  
.....  
.....  
..... [3]

(ii) A program will be translated using a compiler.

Identify the stage of the program development cycle where a syntax error may occur.

..... [1]





(ii) Suggest **two** different validation checks that could be applied to the input of the procedure `GuessNum()` to ensure invalid guesses are not counted.

1 .....

.....

.....

2 .....

.....

.....

[2]

(b) Alice is converting her pseudocode into a high-level language for use in a larger modular program.

She wants to start testing the program before all the subroutines (procedures and functions) have been implemented.

(i) Identify this type of testing.

..... [1]

(ii) Her program contains a function `Status()` that she has not yet written, but will be called from several places within the program.

Explain what Alice needs to do to allow the program to be tested.

.....

.....

.....

.....

..... [2]

(iii) Alice compiles her program.

Explain the function of the compiler.

.....

..... [1]

6 A program stores stock data in four global arrays as follows:

Array	Data type	Description	Example data value	Initial data value
StockID	STRING	The stock item ID (Eight alpha-numeric characters)	"HWDM0001"	""
Description	STRING	A description of the item (Alphabetic characters only)	"Candle"	""
Quantity	INTEGER	The number in stock	4	0
Cost	REAL	The cost of the item	2.75	0

- Each array contains 10000 elements.
- Elements with the same index relate to the same stock item. For example, `StockID[4]` contains the ID for the product whose description is in `Description[4]`.
- The `StockID` array is not sorted and unused elements may occur at any index position.
- Unused elements are assigned the initial data value shown in the table above.

The program is to be modified so that the data from the arrays can be stored in a text file for backup.

The programmer has started to define program modules as follows:

Module	Description
<code>GetValidFilename()</code>	<ul style="list-style-type: none"> <li>• prompts, inputs and returns a valid filename</li> </ul>
<code>Check()</code>	<ul style="list-style-type: none"> <li>• called with an array index as a parameter</li> <li>• checks that the data values for the stock item with the given index are valid</li> <li>• returns <code>TRUE</code> if the values are valid, otherwise returns <code>FALSE</code></li> </ul>
<code>Backup()</code>	<ul style="list-style-type: none"> <li>• calls <code>GetValidFilename()</code> to get the name of the backup file</li> <li>• combines the data values for each stock item to form a single string. Inserts an asterisk character '*' as a separator between data values</li> <li>• writes the string to the backup file</li> <li>• calls <code>Check()</code> to validate the data values. If there is an error then also writes the string to the file "ERRORLOG.TXT"</li> <li>• repeats for all stock items</li> <li>• returns <code>TRUE</code> if nothing was written to "ERRORLOG.TXT", otherwise returns <code>FALSE</code></li> </ul>
<code>Unpack()</code>	<ul style="list-style-type: none"> <li>• called with two parameters: <ul style="list-style-type: none"> <li>○ an array index</li> <li>○ a string value read from one line of the backup file</li> </ul> </li> <li>• extracts the four data values from the string and assigns each to the appropriate array</li> </ul>







.....

.....

.....

.....

.....

.....

..... [8]

(c) The module description of `Unpack()` is repeated here for reference.

Module	Description
<code>Unpack()</code>	<ul style="list-style-type: none"><li>• called with two parameters:<ul style="list-style-type: none"><li>○ an array index</li><li>○ a string value read from one line of the backup file</li></ul></li><li>• extracts the four data values from the string and assigns each to the appropriate array</li></ul>

Write **program code** for the module `Unpack()`. You may assume that the backup file contains only valid stock information.

Visual Basic and Pascal: You should include the declaration statements for variables.  
Python: You should show a comment statement for each variable used with its data type.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....



## Appendix

### Built-in functions (pseudocode)

Each function returns an error if the function call is not properly formed.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER)` RETURNS STRING  
returns a string of length `y` starting at position `x` from `ThisString`

Example: `MID("ABCDEFGH", 2, 3)` returns "BCD"

`LENGTH(ThisString : STRING)` RETURNS INTEGER  
returns the integer value representing the length of `ThisString`

Example: `LENGTH("Happy Days")` returns 10

`LEFT(ThisString : STRING, x : INTEGER)` RETURNS STRING  
returns leftmost `x` characters from `ThisString`

Example: `LEFT("ABCDEFGH", 3)` returns "ABC"

`RIGHT(ThisString : STRING, x : INTEGER)` RETURNS STRING  
returns rightmost `x` characters from `ThisString`

Example: `RIGHT("ABCDEFGH", 3)` returns "FGH"

`INT(x : REAL)` RETURNS INTEGER  
returns the integer part of `x`

Example: `INT(27.5415)` returns 27

`LCASE(ThisChar : CHAR)` RETURNS CHAR  
returns the character value representing the lower case equivalent of `ThisChar`  
If `ThisChar` is not an upper case alphabetic character, it is returned unchanged.

Example: `LCASE('W')` returns 'w'

`ASC(ThisChar : CHAR)` RETURNS INTEGER  
returns the ASCII value of character `ThisChar`

Example: `ASC('A')` returns 65

`RAND(x : INTEGER)` RETURNS REAL  
returns a real number in the range 0 to `x` (not inclusive of `x`).

Example: `RAND(87)` could return 35.43

`NUM_TO_STRING(x : REAL)` RETURNS STRING  
returns a string representation of a numeric value.  
Note: This function will also work if `x` is of type INTEGER

Example: `NUM_TO_STRING(87.5)` returns "87.5"

**Operators (pseudocode)**

<b>Operator</b>	<b>Description</b>
&	Concatenates (joins) two strings Example: "Summer" & " " & "Pudding" produces "Summer Pudding"
AND	Performs a logical AND on two Boolean values Example: TRUE AND FALSE produces FALSE
OR	Performs a logical OR on two Boolean values Example: TRUE OR FALSE produces TRUE

**BLANK PAGE**

---

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge Assessment International Education Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at [www.cambridgeinternational.org](http://www.cambridgeinternational.org) after the live examination series.

Cambridge Assessment International Education is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.