Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER				

COMPUTER SCIENCE **9608/23**

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.

DC (PQ/FC) 205019/2

1 **(a)** Algorithms usually consist of three different types of activity.

Complete the following table.

Write each example statement in **program code and** state the programming language used.

The third activity has already been given.

Activity	Example statement in program code	Programming language
OUTPUT		

[5]

(b) An algorithm searches a 1D array to find the first index of an element that contains a given value. If the value is found, the index of that element is returned.

(i) State an appropriate loop structure for this algorithm. Justify your choice.

Loop structure .....

Justification .....

.....

[2]

(ii) Give **two** possible reasons why the search for the value in **part (b)(i)** would end.

You should assume there is no error.

1 .....

.....

2 .....

.....

[2]

(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 and 23 for the list of built-in pseudocode functions and operators.

Statement	Error
`Code ← RIGHT("Cap" * 3, 2)`	
`Valid ← IS_NUM(3.14159)`	
`NextChar ← MID(ThisString, Index), 1`	

[3]

2 **(a)** After using a program for some time, a user notices a fault in the program.

Describe the term **program fault**.

.....

.....

.....

..... [2]

(b) Good programming practice may help to avoid faults. The use of sensible identifier names is one example of good practice.

(i) Explain the reason for using sensible identifier names.

.....

..... [1]

(ii) State **three other** examples of good programming practice.

1 .....

2 .....

3 .....

[3]

(c) A programmer chooses data to test each path through her program.

Identify the type of testing that the programmer has decided to perform.

..... [1]

3 **(a)** The process of decomposition is often applied to a programming problem.

Describe the process of decomposition.

.....

.....

.....

.....

.....

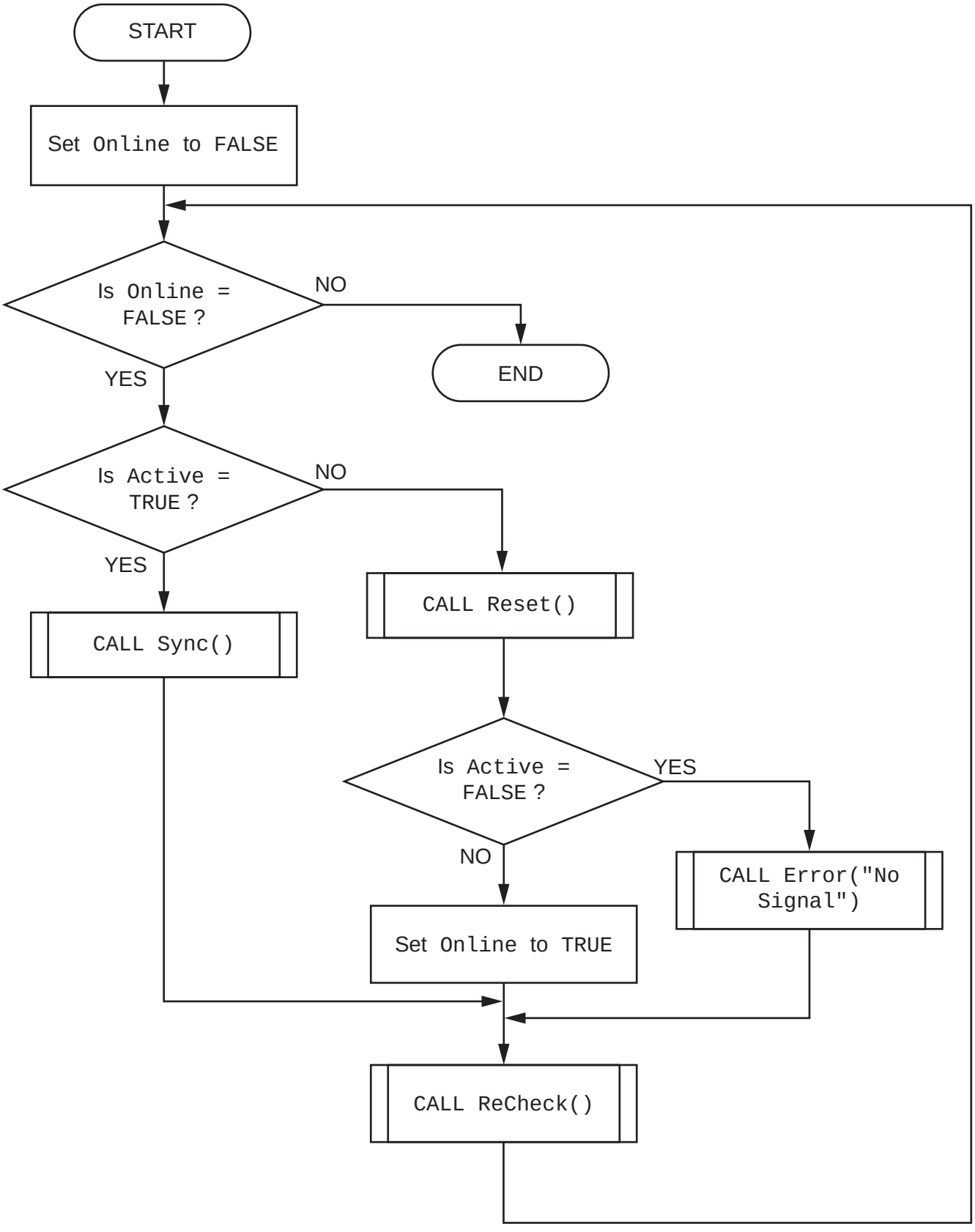
..... [2]

(b) `Result` is a 1D array of type `STRING`. It contains 100 elements.

Draw a program flowchart for an algorithm that will output each element in the array.

[4]

(c) The program flowchart for part of an algorithm from a mobile phone program is shown. Identifier `Active` is a global variable of type `BOOLEAN`.



7

Write **pseudocode** to implement the algorithm represented by the flowchart.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

4 Study the following pseudocode for a string handling function `Check()`.

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

```
FUNCTION Check(InString : STRING) RETURNS INTEGER

DECLARE Index, Result, Count : INTEGER
DECLARE NextChar : CHAR

   Result ← 0
   Count ← 1

   FOR Index ← 1 TO LENGTH(InString)

      NextChar ← MID(InString, Index, 1)
      IF (NextChar >= '0' AND NextChar <= '9') OR NextChar = '.'
         THEN
            Result ← Result + 1
         ELSE
            IF NextChar = ','
               THEN
                  Count ← Count + 1
               ELSE
                  Result ← -1
            ENDIF
      ENDIF
   ENDFOR

   IF Count < 3
      THEN
         RETURN -1
      ELSE
         RETURN Result
   ENDIF


ENDFUNCTION
```

(a) (i) Complete the trace table by performing a dry run of the function when it is called as follows:

```
Answer ← Check("74.0,4.6,3x2")
```

Note, there are no space characters in the string shown.

Result	Count	Index	NextChar

[5]

(ii) State the value returned by the function when it is called as shown in **part (a)(i)**.

..... [1]

(b) A number group is a string of characters that represents an integer or decimal value. A comma separates number groups.

For example, `"74.0"` is a number group in the string `"74.0,4.6,3x2"`.

The function `Check()` is intended to analyse the number groups in the parameter passed.

The function returns:

- a count of the number groups in the parameter passed

or

- −1 if there are less than three number groups in the string **or** if any non-numeric characters occur in the string (other than decimal point and comma).

There is an error in the algorithm causing an incorrect value to be returned by the function.

(i) Explain why this error can occur.

.....

.....

.....

..... [2]

(ii) Describe how the algorithm could be amended to correct the error.

.....

.....

.....

..... [1]

(c) A dry run of a pseudocode algorithm may help to locate logic errors.

Give **another** type of program error and describe how it can occur.

Type of error .....

Description .....

.....

.....

.....

[2]

BLANK PAGE

[Turn over

5 A program stores a contact list of telephone numbers. Each telephone number is stored as a string of six or more numeric characters.

Before they are displayed, number strings are formatted to make them easier to read. This involves forming the characters into groups, separated by the space character.

The maximum length of a number group is five characters.

Different numbers may have different groupings. A template string is used to define the grouping.

For example:

Number string	Template string	Formatted string
`"01223553998"`	`"53"`	`"01223 553 998"`
`"509700101"`	`"222"`	`"50 97 00 101"`
`"4044496128"`	`"33"`	`"404 449 6128"`

For the first row, template `"53"` results in a formatted string comprising:

- the first five characters in the first group
- a space character
- the next three characters in the second group
- a space character
- the remaining characters from the number string.

(a) Write **pseudocode** for a function `GroupNum()`, which takes a telephone number and a template as parameter strings and returns a formatted string.

You may assume that the template and telephone number are valid.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[8]

(b) The function `GroupNum()` is to be extended to include parameter checking.

State **one** check that could be applied to **each** parameter.
Give an example of test data that could be used to demonstrate that each check identifies invalid data.

The type of check must be different for each parameter.

Telephone number check .....

.....

.....

.....

Test data .....

Template check .....

.....

.....

.....

Test data .....

[4]

BLANK PAGE

6 A program stores data about stock items in four global 1D arrays as follows:

Array	Data type	Description	Example data value	Initial data value
`StockID`	`STRING`	the stock item ID (eight alpha-numeric characters)	`"ABLK0001"`	`""`
`Description`	`STRING`	a description of the item (alphabetic characters only)	`"torch"`	`""`
`Quantity`	`INTEGER`	the number in stock	`6`	`0`
`Cost`	`REAL`	the cost of the item	`4.80`	`0.0`

- Each array contains 10 000 elements.
- Elements with the same index relate to the same stock item. For example, `StockID[5]` contains the ID for the product whose description is in `Description[5]`.
- 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 first four characters of the `StockID` represent a product group. The last four characters represent the item within the group.

The program is to be modified so that:

- data from the arrays are stored in a text file for backup purposes. Data from unused elements are not stored in the file.
- a `Summary` array is added. This will be a global 1D array of 500 elements of type `STRING`. Each product group will occur **once** in the array, for example `"ABLK"` for the item in the table above.

The programmer has started to define program modules as follows:

Module	Description
`GetValidFilename()`	• prompts and inputs a filename • returns a valid filename as a `STRING`
`CheckBackupFile()`	• calls `GetValidFilename()` for a filename • checks if the file is empty • If the file is not empty ask the user to confirm that overwrite is intended. If not intended allow the user to re-input a different filename. • returns the filename.

(a) Write **pseudocode** for the module `CheckBackupFile()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(b) Write **program code** for a module `GroupReport()`, which will summarise the stock data for a given product group.

The product group will be passed to the module as a string. The total value of items is calculated by multiplying the cost by the quantity.

An example of the output for group ABLK is as follows:

```
Group: ABLK
Number of items in Group: 11
Total value of items in Group: 387.89
```

If no items are found for group ABLK, the output is as follows:

```
There are no items in Group: ABLK
```

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(c) Two additional modules are required:

Module	Description
`Lookup()`	• called with a `STRING` representing a product group (for example, `"ABLK"`) • searches the `Summary` array for the group • returns the index position or returns −1 if not found
`GroupSummary()`	• stores each product group name (found in the `StockID` array) into the `Summary` array, if not there already • calls `Lookup()` to check whether the name is already in the `Summary` array • returns the number of product groups added to the `Summary` array

You can assume:

- all elements of the `Summary` array have been initialised to the value `""` before `GroupSummary()` is called
- there will be no more than 500 product groups.

Write **program code** for the module `GroupSummary()`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

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`

`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"`

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

Example: `STRING_TO_NUM("23.45")` returns `23.45`

`IS_NUM(ThisString : STRING) RETURNS BOOLEAN`
returns the value `TRUE` if `ThisString` contains only numeric characters (`'0'` to `'9'`).

Example: `IS_NUM("123a")` returns `FALSE`

Operators (pseudocode)

Operator	Description
`&`	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