Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2019**

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.
You may use an HB pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.
No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **18** printed pages and **2** blank pages.

DC (ST) 163547/2

Cambridge Assessment International Education

[Turn over

1 The following pseudocode searches for the longest run of identical characters in the array `Message`.

```
DECLARE Message : ARRAY[1:100] OF CHAR


PROCEDURE Search()

   DECLARE Index : INTEGER
   DECLARE ThisChar : CHAR
   DECLARE ThisRun : INTEGER
   DECLARE LongRun : INTEGER

   ThisChar ← Message[1]
   ThisRun ← 1
   LongRun ← 1

   FOR Index ← 2 TO 100

      IF Message[Index] = ThisChar
         THEN
            ThisRun ← ThisRun + 1
         ELSE
            ThisChar ← Message[Index]
            IF ThisRun > LongRun
               THEN
                  LongRun ← ThisRun
            ENDIF
            ThisRun ← 1
      ENDIF

   ENDFOR

   OUTPUT "The longest run was " , LongRun

ENDPROCEDURE
```

(a) Draw a program flowchart to represent the procedure `Search()`.

Variable and array declarations are not required in program flowcharts.

[6]

(b) (i) Program variables have values as follows:

Variable	Value
MeltingPoint	180.5
Soluble	FALSE
Attempt	3
ProductName	"Mushroom Compost"
Version	'A'
ProductID	"BZ27-4"

Evaluate each expression in the following table.

If an expression is invalid, write ERROR.

For the built-in functions list, refer to the **Appendix** on page 18.

Expression	Evaluates to
STRING_TO_NUM(MID(ProductID, 3, 2)) + 4	
INT(MeltingPoint / 2)	
Soluble AND Attempt > 3	
LENGTH(ProductID & NUM_TO_STRING(MeltingPoint))	
RIGHT(ProductName, 4) & MID(ProductName, 5, 4)	

[5]

(ii) Programming languages support different data types.

Give an appropriate data type for the following variables from **part (b)(i)**.

Variable	Data type
MeltingPoint	
Soluble	
Attempt	
Version	
ProductID	

[5]

2 **(a)** A student is learning about arrays. She wants to write a program to:

- search through a 1D array of 100 elements
- count the number of elements that contain the string “Empty”
- output the number of elements containing “Empty” together with a suitable message.

Use **structured English** to describe the algorithm she could use.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) She uses the process of stepwise refinement to develop her algorithm.

Explain this process.

.....

.....

.....

.....

.....

..... [3]

[Turn over

(c) The student is learning about file handling.

She has been told that there are different file modes that can be used when opening a text file. She wants to make sure that the existing contents are not deleted when the file is opened.

Identify **two** file modes she could use **and** describe their use.

Mode .....

Description .....

.....

.....

Mode .....

Description .....

.....

.....

[4]

(d) The student has completed the design of her program and is ready to use an Integrated Development Environment (IDE).

Describe the features of an IDE that she can use to write, translate and test her program.

.....

.....

.....

.....

.....

..... [3]

Question 3 begins on the next page.

3 The following pseudocode represents three separate modules from an algorithm design. The module contents are not shown.

```
FUNCTION Search(AA : INTEGER, BB : STRING) RETURNS INTEGER
  /
ENDFUNCTION


FUNCTION Allocate() RETURNS BOOLEAN
  /
ENDFUNCTION


PROCEDURE Enable(CC : INTEGER, BYREF DD : INTEGER)
  /
ENDPROCEDURE
```

A fourth module, `Setup()`, refers to the previous three modules as follows:

```
PROCEDURE Setup()
/
  WHILE Authorised = TRUE
     /
     ThisValue ← Search(27, "Thursday")
     /
     Authorised ← Allocate()
     /
     CALL Enable(ThisValue, 4)
     /
  ENDWHILE
/
ENDPROCEDURE
```

(a) Draw a structure chart to show the four modules and the parameters that these pass between them.

[6]

(b) The algorithm is implemented in a high-level language. Changes are required and the program is given to Albert, who is an experienced programmer. He is not familiar with the language that has been used.

Explain why Albert would be able to understand the program.

.....

.....

.....

..... [2]

4 A program is being written to process student information. One task involves inputting the names of all students in a class.

A first attempt at the pseudocode for this task is as follows:

```
DECLARE Name1 : STRING
DECLARE Name2 : STRING
DECLARE Name3 : STRING
```

```
DECLARE Name40 : STRING

OUTPUT "Input the name for student 1"
INPUT Name1
OUTPUT "Input the name for student 2"
INPUT Name2
OUTPUT "Input the name for student 3"
INPUT Name3
```

```
OUTPUT "Input the name for student 40"
INPUT Name40
```

(a) Re-write the **pseudocode** to perform this task in a more efficient way, to allow for the class of 40 students.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) Give **one** advantage of your solution.

..... [1]

Question 5 begins on the next page.

5 Nigel is learning about string handling. He wants to write code to count the number of words in a given string. A word is defined as a sequence of alphabetic characters that is separated by one or more space characters.

His first attempt at writing an algorithm in pseudocode is as follows:

```
PROCEDURE CountWords(Message : STRING)

   DECLARE NumWords : INTEGER
   DECLARE Index : INTEGER
   CONSTANT Space = ' '

   NumWords ← 0

   FOR Index ← 1 TO LENGTH(Message)
      IF MID(Message, Index, 1) = Space
         THEN
            NumWords ← NumWords + 1
      ENDIF
   ENDFOR

   OUTPUT "Number of words : " , NumWords

ENDPROCEDURE
```

For the built-in functions list, refer to the **Appendix** on page 18.

His first attempt is incorrect. He will use white-box testing to help him to identify the problem.

(a) (i) State the purpose of white-box testing.

.....

..... [1]

(ii) Dry running the code is often used in white-box testing. In this method, the programmer records the values of variables as they change.

Identify what the programmer would normally use to record the changes.

..... [1]

(b) (i) Write a test string containing two words that gives the output:

```
Number of words : 2
```

Use the symbol `'∇'` to represent each space character in your test string.

Explain why the algorithm gives the output shown above.

String .....

Explanation .....

.....

.....

.....

.....

[3]

(ii) Nigel tested the procedure with the strings:

String 1: `"Red∇and∇Yellow"`
String 2: `"Green∇∇and∇∇Pink∇"`

Give the output that is produced for each of the strings.

Describe the changes that would need to be made to the algorithm to give the correct output in each case.

Do **not** write pseudocode **or** program code.

String 1 .....

Description .....

.....

.....

.....

String 2 .....

Description .....

.....

.....

.....

[6]

6 A text file, `StudentContact.txt`, contains a list of names and telephone numbers of students in a school. Not all students in the school have provided a contact telephone number. In this case, their name will not be in the file.

Each line of the file is stored as a string that contains a name and telephone number, separated by the asterisk character (`'*'`) as follows:

`<Name>'*'<TelNumber>`, for example:

`"Bill Smith*081234567"`

A 1D array, `ClassList`, contains the names of students in a particular class. The array consists of 40 elements of string data type. You can assume that student names are unique.
Unused elements contain the empty string `""`.

A program is to be written to produce a **new** text file, `ClassContact.txt`, containing student names and numbers for all students in a particular class.

For each name contained in the `ClassList` array, the program will:

- search the `StudentContact.txt` file
- copy the matching string into `ClassContact.txt` if the name is found
- write the name together with "*No number" into `ClassContact.txt` if the name is not found.

The program will be implemented as three modules. The description of these is as follows:

Module	Description
`ProcessArray()`	• Check each element of the array: ○ Read the student name from the array ○ Ignore unused elements ○ Call `SearchFile()` with the student name ○ If the student name is found, call `AddToFile()` to write the student details to the class file ○ If the student name is not found, call `AddToFile()` to write a new string to the class file, formed as follows: <Name>"*No number" • Return the number of students who have not provided a telephone number
`SearchFile()`	• Search for a given student name at the start of each line in the file `StudentContact.txt`: ○ If the search string is found, return the text line from `StudentContact.txt` ○ If the search string is not found, return an empty string
`AddToFile()`	• Append the given string to a specified file, for example, `AddToFile(StringName, FileName)`

(a) Write **program code** for the module `SearchFile()`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(b) Write **pseudocode** for the module `ProcessArray()`.

.... [9]

(c) `ProcessArray()` is modified to make it general purpose. It will now be called with two parameters as follows:

- an array
- a string representing the name of a class contact file

It will still return the number of students who have not provided a contact telephone number.

Write **program code** for the header (declaration) of the modified `ProcessArray()`.

Programming language .....

Program code

.....

.....

.....

..... [3]

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.

Example: `NUM_TO_STRING(x)` returns `"87.5"` if `x` has the value `87.5`
Note: This function will also work if `x` is of type `INTEGER`

`STRING_TO_NUM(x : STRING) RETURNS REAL`
returns a numeric representation of a string.

Example: `STRING_TO_NUM(x)` returns `23.45` if `x` has the value `"23.45"`
Note: This function will also work if `x` is of type `CHAR`

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

BLANK PAGE