Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME

CENTRE NUMBER | | | | | |

CANDIDATE NUMBER | | | | |

9744681161

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 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 **17** printed pages and **3** blank pages.

Cambridge Assessment International Education

1 Study the following pseudocode.

```
FUNCTION Search() RETURNS INTEGER
   DECLARE N, C : INTEGER
   DECLARE V, L : REAL
   V ← GetLevel()
   L ← V * 1.34
   C ← 0
   FOR N ← 1 TO 10
      V ← GetLevel()
      IF V > L
         THEN
            C ← C + 1
      ENDIF
   ENDFOR
   OUTPUT "Process complete"
   RETURN C
ENDFUNCTION
```

(a) (i) This pseudocode lacks features that would make it easier to read and understand.

State **three** such features.

Feature 1 .....

Feature 2 .....

Feature 3 .....

[3]

(ii) Draw a program flowchart to represent the algorithm implemented in the pseudocode. Variable declarations are not required in program flowcharts.

[5]

[Turn over

(b) (i) Programming languages support different data types.

Complete the table by giving a suitable data type for each example value.

Example value	Data type
`"NOT TRUE"`	
`-4.5`	
`NOT FALSE`	
`132`	

[4]

(ii) Evaluate each expression in the following table.

If an expression is invalid then write 'ERROR'.

Refer to the **Appendix** on page 16–17 for the list of built-in functions and operators.

Expression	Evaluates to
`LEFT("Start", 3) & RIGHT("Apple", 3)`	
`MID("sample", 3, 5)`	
`NUM_TO_STRING(12.3 * 2)`	
`INT(STRING_TO_NUM("53.4")) + 7`	

[4]

2 **(a)** A structure chart is often used in modular program design. One feature shown is the sequence of module execution.

State **four** other features that may be shown.

Feature 1 .....

.....

Feature 2 .....

.....

Feature 3 .....

.....

Feature 4 .....

.....

[4]

(b) Identify and describe **one** feature of an Integrated Development Environment (IDE) that can help with **program presentation**.

Feature .....

Description .....

.....

[2]

(c) **By value** is one method of passing a parameter to a subroutine.

Identify and describe the other method.

Method .....

Description .....

.....

.....

[2]

(d) Explain the term **adaptive maintenance**.

.....

.....

.....

..... [2]

3 The following is a function design in pseudocode.

Line numbers are given for reference only.

```
10 FUNCTION Check(InString : STRING) RETURNS BOOLEAN
11
12    DECLARE NumDots : INTEGER
13    DECLARE Index : INTEGER
14    DECLARE NumOthers : INTEGER
15
16    NumDots ← 0
17    NumOthers ← 0
18    Index ← 1
19
20    WHILE NumDots < 3 AND Index <= LENGTH(InString)
21
22       IF MID(InString, Index, 1) = '.'
23          THEN
24             NumDots ← NumDots + 1
25          ELSE
26             NumOthers ← NumOthers + 1
27       ENDIF
28       Index ← Index + 1
29
30    ENDWHILE
31
32    IF NumDots = NumOthers
33       THEN
34          RETURN TRUE
35       ELSE
36          RETURN FALSE
37    ENDIF
38
39 ENDFUNCTION
```

Study the pseudocode. Identify the relevant features in the following table.

Refer to the **Appendix** on pages 16–17 for the list of built-in functions and operators.

Feature	Answer
The number of the line containing a variable being incremented	
The range of line numbers containing a pre-condition loop	
The number of initialisation statements	
The number of the line containing a logical operator	
The range of line numbers containing a selection statement	
The name of a built-in function	
The name of a parameter	

[7]

4 A student is developing a program to count how many times each character of the alphabet (A to Z) occurs in a given string. Upper case and lower case characters will be counted as the same. The string may contain non-alphabetic characters, which should be ignored.

The program will:

- check each character in the string to count how many times each alphabetic character occurs
- store the count for each alphabetic character in a 1D array
- output each count together with the corresponding character.

(a) The student has written a structured English description of the algorithm:

1. START at the beginning of the string
2. SELECT a character from the string
3. CONVERT the character to upper case
4. CHECK whether the character is alphabetic and INCREMENT as required.
5. REPEAT from step 2 until last character has been checked
6. OUTPUT a suitable message giving the count of each alphabetic character

Step 4 above is not described in sufficient detail.

The student decides to apply a process to increase the level of detail given in step 4.

State the name of the process **and** use this process to write step 4 in more detail. Use **structured English** for your answer.

Process .....

Structured English .....

.....

.....

.....

.....

.....

[4]

(b) Write **pseudocode** to implement the program.

You should note the following:

- `InString` contains the string to be checked. It has been assigned a value.
- The elements of the array `Result` have all been initialised to zero.
- The ASCII value of letter 'A' is 65.

You should assume the following lines of pseudocode have been written:

```
DECLARE InString : STRING
DECLARE Result : ARRAY [1:26] OF INTEGER
```

Declare any further variables you use. Do **not** implement the code as a subroutine.

Refer to the **Appendix** on pages 16–17 for the list of built-in functions and operators.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

5 The following pseudocode checks whether a string is a valid password.

```
FUNCTION CheckPassword(InString : STRING) RETURNS BOOLEAN

   DECLARE Index, Upper, Lower, Digit, Other : INTEGER
   DECLARE NextChar : CHAR

   Upper ← 0
   Lower ← 0
   Digit ← 0
   Other ← 0

   FOR Index ← 1 TO LENGTH(InString)

      NextChar ← MID(InString, Index, 1)
      IF NextChar >= 'A' AND NextChar <= 'Z'
         THEN
            Upper ← Upper + 1
         ELSE
         IF NextChar >= 'a' AND NextChar <= 'z'
            THEN
               Lower ← Lower + 1
            ELSE
               IF NextChar >= '0' AND NextChar <= '9'
                  THEN
                     Digit ← Digit + 1
                  ELSE
                     Other ← Other + 1
               ENDIF
         ENDIF
      ENDIF

   ENDFOR

   IF Upper > 1 AND Lower >= 5 AND (Digit - Other) > 0
      THEN
         RETURN TRUE
      ELSE
         RETURN FALSE
   ENDIF

ENDFUNCTION
```

(a) Describe the validation rules that are implemented by this pseudocode. Refer **only** to the contents of the string and **not** to features of the pseudocode.

.....

.....

.....

.....

.....

..... [3]

(b) (i) Complete the trace table by dry running the function when it is called as follows:

```
Result ← CheckPassword("Jim+Smith*99")
```

Index	NextChar	Upper	Lower	Digit	Other

[5]

(ii) State the value returned when the function is called using the expression shown. Justify your answer.

Value .....

Justification .....

.....

.....

[2]

6 Account information for users of a library is held in one of two text files; `UserListAtoM.txt` and `UserListNtoZ.txt`

The format of the data held in the two files is identical. Each line of the file is stored as a string that contains an account number, name and telephone number separated by the asterisk character (`'*'`) as follows:

```
<Account Number>'*'<Name>'*'<Telephone Number>
```

An example of one line from the file is:

```
"GB1234*Kevin Mapunga*07789123456"
```

The account number string may be **six** or **nine** characters in length and is **unique for each person**. It is made up of alphabetic and numeric characters only.

An error has occurred and the same account number has been given to different users in the two files. There is **no** duplication of account numbers **within each individual file**.

A program is to be written to search the two files and to identify duplicate entries. The account number of any duplicate found is to be written to an array, `Duplicates`, which is a 1D array of 100 elements of data type `STRING`.

The program is to be implemented as several modules. The outline description of three of these is as follows:

Module	Outline description
`ClearArray()`	• Initialise the global array `Duplicates`. Set all elements to the empty string.

`FindDuplicates()`	• Read each line from the file `UserListAtoM.txt`
	  ◦ Check whether the account number appears in file `UserListNtoZ.txt` using `SearchFileNtoZ()`
	  ◦ If the account number does appear then add the account number to the array.
	• Output an error message and exit the module if there are more duplicates than can be written to the array.

`SearchFileNtoZ()`	• Search for a given account number in file `UserListNtoZ.txt` ◦ If found, return `TRUE`, otherwise return `FALSE`

(a) State **one** reason for storing data in a file rather than in an array.

.....

..... [1]

(b) Write **program code** for the module `SearchFileNtoZ()`.

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]

(c) Write **pseudocode** for the module `FindDuplicates()`.

The module description is given in the table on page 12.

.....

.....

.....

.....

.....

.....

..... [8]

(d) `ClearArray()` is to be modified to make it general purpose. It will be used to initialise any 1D array of data type `STRING` to any value.

It will now be called with three parameters as follows:

1. The array
2. The number of elements
3. The initialisation string

You should assume that the lower bound is 1.

(i) Write **pseudocode** for the modified `ClearArray()` procedure.

.....

.....

.....

.....

.....

.....

.....

..... [3]

(ii) Write **program code** for a statement that calls the modified `ClearArray()` procedure to clear the array `Duplicates` to `"Empty"`.

Programming language .....

Program code

.....

.....
[2]

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`

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

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

`CHR(x : INTEGER) RETURNS CHAR`
returns the character whose ASCII value is `x`

Example: `CHR(87)` returns `'W'`

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

Example: `UCASE('a')` returns `'A'`

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

BLANK PAGE