

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/22

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 **18** printed pages and **2** blank pages.

1 Study the following pseudocode.

```

PROCEDURE FillTank()

    DECLARE Tries : INTEGER
    DECLARE Full : BOOLEAN

    Tries ← 1

    Full ← ReadSensor("F1")

    IF NOT Full
        THEN
            WHILE NOT Full AND Tries < 4
                CALL TopUp()
                Full ← ReadSensor("F1")
                Tries ← Tries + 1
            ENDWHILE
            IF Tries > 3
                THEN
                    OUTPUT "Too many attempts"
                ELSE
                    OUTPUT "Tank now full"
                ENDIF
            ELSE
                OUTPUT "Already full"
            ENDIF
        ENDIF
    ENDPROCEDURE

```

- (a) (i) The pseudocode includes features that 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]

- (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
43	
TRUE	
-273.16	
"-273.16"	

[4]

- (ii) Evaluate each expression in the following table.

If an expression is invalid then write 'ERROR'.

Refer to the **Appendix** on page 18 for the list of built-in functions and operators.

Expression	Evaluates to
RIGHT("Stop", 3) & LEFT("ich", 2)	
MID(NUM_TO_STRING(2019), 3, 1)	
INT(NUM_TO_STRING(-273.16))	
INT(13/2)	

[4]

2 (a) Describe the program development cycle with reference to the following:

- source code
- object code
- corrective maintenance.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

(b) Give **three** features of an Integrated Development Environment (IDE) that can help with **initial error detection** while writing the program.

1

.....

2

.....

3

.....

..... [3]

- 3 A student is developing a program to search through a string of numeric digits to count how many times each digit occurs. The variable `InString` will store the string and the 1D array `Result` will store the count values.

The program will:

- check each character in the string to count how many times each digit occurs
- record the count for each digit using the array
- output the count for each element of the array together with the corresponding digit.

- (a) The array `Result` is a 1D array of type `INTEGER`.

Write **pseudocode** to declare the array and to initialise all elements to zero.

.....

.....

.....

.....

.....

.....

..... [3]

(b) Write the **pseudocode** for the program.

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

Refer to the **Appendix** on page 18 for the list of built-in functions and operators.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

[8]

- 4 A program is being written to control the operation of a portable music player. One part of the program controls the output volume.

The player has two buttons, one to increase the volume and one to decrease it. Whenever a button is pressed, a procedure `Button()` is called with a parameter value representing the button as follows:

Button	Parameter value
Volume increase	10
Volume decrease	20

For example, pressing the volume increase button three times followed by pressing the volume decrease button once would result in the calls:

```
CALL Button(10) // VolLevel increased by 1
CALL Button(10) // VolLevel increased by 1
CALL Button(10) // VolLevel increased by 1
CALL Button(20) // VolLevel decreased by 1
```

The program makes use of two global variables of type `INTEGER` as follows:

Variable	Description
<code>VolLevel</code>	The current volume setting. This must be in the range 0 to 49.
<code>MaxVol</code>	A value that can be set to limit the maximum value of <code>VolLevel</code> , in order to protect the user's hearing. A value in the range 1 to 49 indicates the volume limit. A value of zero indicates that no volume limit has been set.

The procedure `Button()` will modify the value of `VolLevel` depending on which button has been pressed and whether a maximum value has been set.

(b) The procedure `Button()` is to be tested using black-box testing.

Fill in the gaps below to define **three** tests that could be carried out.

TEST 1 – `VolLevel` is changed

Parameter value: 10

`MaxVol`:

`VolLevel` value before call to `Button()`: 48

`VolLevel` expected value after call to `Button()`:

TEST 2 – `VolLevel` is **not** changed

Parameter value: 10

`MaxVol`: 34

`VolLevel` value before call to `Button()`:

`VolLevel` expected value after call to `Button()`:

TEST 3 – `VolLevel` is **not** changed

Parameter value:

`MaxVol`: 40

`VolLevel` value before call to `Button()`: 0

`VolLevel` expected value after call to `Button()`:

[6]

(c) The testing stage is part of the program development cycle.

(i) The program for the music player has been completed. The program does not contain any syntax errors, but testing could reveal further errors.

Identify **and** describe **one different** type of error that testing could reveal.

Type

Description

.....

.....

[2]

(ii) Stub testing is a technique often used in the development of modular programs.

Describe the technique.

.....

.....

.....

.....

.....

..... [3]

- 5 The module headers for three modules in a program are defined in pseudocode as follows:

Pseudocode module header
PROCEDURE Lookup(P4 : INTEGER, BYREF M4 : STRING)
FUNCTION Update(T4 : INTEGER) RETURNS INTEGER
FUNCTION Validate(S2 : INTEGER, P3 : STRING) RETURNS BOOLEAN

A fourth module, `Renew()`, calls the three modules in the following sequence.

```
Validate()  
Lookup()  
Update()
```

Draw a structure chart to show the relationship between the four modules and the parameters passed between them.



[7]

Question 6 begins on the next page.

6 A text file, `StudentList.txt`, contains a list of information about students in a school.

Each line of the file contains a reference, name and date of birth for one student. All the information is held as strings and separated by the asterisk character ('*') as follows:

```
<Reference>' * '<Name>' * '<Date Of Birth>
```

An example of one line from the file is:

```
"G1234*Aleza Hilton*05062001"
```

The reference string may be five or eight characters in length and is unique for each student. It is made up of alphabetic and numeric characters only.

A global 1D array, `Leavers`, contains the references of all students who have recently left the school. The array consists of 500 elements of data type `STRING`. Unused elements contain the empty string "".

A program is to be written to produce a new text file, `UpdatedList.txt`, containing information only for students who are still attending the school.

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

Module	Outline description
<code>ProcessStudentList()</code>	<ul style="list-style-type: none"> • Read each line from the file <code>StudentList.txt</code> <ul style="list-style-type: none"> ◦ Check whether the <code>Reference</code> appears in the array using <code>SearchLeavers()</code> ◦ If the <code>Reference</code> does not appear then write the line to the file <code>UpdatedList.txt</code> • Return the number of lines not copied.
<code>SearchLeavers()</code>	<ul style="list-style-type: none"> • Search for a given <code>Reference</code> in the array <code>Leavers</code> • If the <code>Reference</code> is found, return <code>TRUE</code>, otherwise return <code>FALSE</code>
<code>CountTimes()</code>	<ul style="list-style-type: none"> • Take two parameters: the name of an array and a string. • Count the number of elements that are the same as the string. Return the count value.

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

The module description is repeated here for reference.

Module	Outline description
<code>ProcessStudentList()</code>	<ul style="list-style-type: none"> • Read each line from the file <code>StudentList.txt</code> <ul style="list-style-type: none"> ◦ Check whether the <code>Reference</code> appears in the array using <code>SearchLeavers()</code> ◦ If the <code>Reference</code> does not appear then write the line to the file <code>UpdatedList.txt</code> • Return the number of lines not copied.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [9]

(c) `CountTimes()` is to be used to count how many unused elements there are in the `Leavers` array. An unused element is one that contains an empty string.

The module description is repeated here for reference.

Module	Outline description
<code>CountTimes()</code>	<ul style="list-style-type: none">• Take two parameters: the name of an array and a string.• Count the number of elements that are the same as the string. Return the count value.

Write a statement in **program code** that uses `CountTimes()` to assign the count of unused elements to the variable `Result`.

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.
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

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

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 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.