Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 9 0 6 6 0 7 6 3 4 2 *

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2015**

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 **14** printed pages and **2** blank pages.

DC (LK/JG) 95389/4
© UCLES 2015

CAMBRIDGE International Examinations

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

1 Horses are entered for a horse race. A horse may have to carry a penalty weight in addition to the rider. This weight is added to the saddle. The penalty weight (if any) depends on the number of wins the horse has achieved in previous races.

The penalty weight is calculated as follows:

Number of previous wins	Penalty weight (kg)
0	0
1 or 2	4
Over 2	8

A program is to be written from the following structured English design.

1 INPUT name of horse

2 INPUT number of previous wins

3 CALCULATE penalty weight

4 STORE penalty weight

5 OUTPUT name of horse, penalty weight

(a) Complete the identifier table showing the variables needed to code the program.

Identifier	Data type	Description

[3]

(b) Line 3 in the algorithm above does not give the detail about how the race penalty weight is calculated; this step in the algorithm must be expressed in more detail.

(i) The algorithm above currently has five stages. One technique for program design is to further break down, where required, any stage to a level of detail from which the program code can be written.

Name this technique.

.....[1]

(ii) Write **pseudocode** for the given structured English design.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

2 **(a)** Two operators available in a programming language are `DIV` and `MOD`. They perform integer arithmetic as follows:

Expression	Explanation
`X DIV Y`	Computes the number of times `Y` divides into `X`
`X MOD Y`	Computes the remainder when `X` is divided by `Y`

Calculate the value of the variables shown for the following code fragments.

	Code	Variable	
(i)	`NumberLeftOver ← 37 MOD 10`	`NumberLeftOver`	[1]

(ii)	`Quantity ← 208`
	`BoxSize ← 100`
	`NumberOfBoxes ← Quantity DIV BoxSize`
	`Temp ← (Quantity MOD BoxSize) + 1`
		`NumberOfBoxes` .....
		`Temp` .....
			[2]

(b) Bank customers withdraw money from their account at a cash dispenser machine using their bank card. The machine operates as follows:

- it can dispense the following notes:
  - $50
  - $20
  - $10
- the maximum amount for a single withdrawal is $500

When a customer withdraws money, they enter the amount to withdraw. (This must be a multiple of $10).

The machine will always dispense the least possible number of notes.

A program is designed for the machine to process a withdrawal.

The following variables are used:

Identifier	Data type	Description
`Amount`	`INTEGER`	Amount to withdraw entered by the user
`FiftyDollar`	`INTEGER`	Number of $50 notes to dispense
`TwentyDollar`	`INTEGER`	Number of $20 notes to dispense
`TenDollar`	`INTEGER`	Number of $10 notes to dispense
`Temp`	`INTEGER`	Used in the calculation of the number of each note required

(i) The following four tests have been designed.

Complete the test data table showing the expected results with comments.

Input value	Output			Comment
Amount	**FiftyDollar**	**TwentyDollar**	**TenDollar**	
70	1	1	0	Least possible number of notes
85				
130				
600				

[3]

(ii) Complete the pseudocode.

```
INPUT ..................................................................................
IF Amount > 500
   THEN
      OUTPUT "Refused – amount too large"
   ELSE
      ..................................................................................
         THEN
            OUTPUT "Refused - not a multiple of $10"
         ELSE
            FiftyDollar ← Amount DIV 50
            Temp ← ...........................................................................
            TwentyDollar ← ...................................................................
            Temp ← ...........................................................................
            ..................................................................................
      ENDIF
ENDIF
```

[5]

3 A flooring company provides for each customer an estimated price for a new job. Each job is given a Job ID.

The job cost is calculated from the length (nearest metre) and width (nearest metre) of the room.

The process for calculating the price is as follows:

- the floor area is calculated with 18% added to allow for wastage
- the job cost is calculated at $50 per square metre

The structure chart shows the modular design for a program to produce a new job cost.

(i) Give the data items corresponding to the labels A to E in the structure chart.

A .....

B .....

C .....

D .....

E .....

[5]

(ii) The procedure below is one of the modules shown on the structure chart.

Parameters can be passed ‘by value’ or ‘by reference’.

Complete the procedure header below showing for each parameter:

- its parameter passing mechanism
- its identifier
- its data type

```
PROCEDURE CalculateJobCost( ..................................................................
                            ..................................................................
                            ..................................................................
                            .................................................................. )
  JobCost ← (Length * Width * 1.18) * 50
ENDPROCEDURE
```

[5]

4 A programming language has the built-in function `CONCAT` defined as follows:

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING] )  RETURNS STRING
For example:  CONCAT("San", "Francisco") returns "SanFrancisco"
              CONCAT("New", "York", "City") returns "NewYorkCity"
```

The use of the square brackets indicates that the parameter is optional.

(a) State the value returned by the following expressions.

If the expression is not properly formed, write ERROR.

(i) `CONCAT("Studio", 54)` ..... [1]

(ii) `CONCAT("parity", "error", "check")` ..... [1]

(iii) `CONCAT(CONCAT("Binary", "▼", "Coded"), "▼", "Decimal")`

▼ indicates a <Space> character

.....[2]

(b) A country has a number of banks. There are cash dispensers all over the country. Each bank is responsible for a number of dispensers.

- banks have a three digit code in the range 001 – 999
- each dispenser has a five digit code in the range 00001 – 99999

A text file, `DISPENSERS`, is to be created.

It has one line of text for each dispenser. For example: `00342▼007`.

This line in the file is the data for dispenser `00342` which belongs to bank `007`.

Incomplete pseudocode follows for the creation of the file `DISPENSERS`.

For the creation of the file, data is entered by the user at the keyboard.

(i) Complete the **pseudocode**.

```
OPENFILE ..................................................... FOR WRITE

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

   OUTPUT "Enter dispenser code (XXXXX to end)"
   INPUT DispenserCode
   IF DispenserCode <> "XXXXX"
      THEN
         OUTPUT "Enter bank code"
         INPUT BankCode
         LineString ← CONCAT(.........................................,"▼", BankCode)
         // now write the new line to the file

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

   ENDIF

UNTIL ........................................................................................

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

OUTPUT "DISPENSERS file now created"                                          [6]
```

(ii) No attempt has been made to validate the data entered by the user.

Describe **two** different types of validation check for the data entry.

1 .....

.....

2 .....

.....[2]

(iii) The programmer coded this algorithm above and the user successfully entered 15 dispenser records into the text file.

There is data for another 546 dispensers which needs to be added.

State the error that will occur if the user runs the program a second time for further data entry.

.....[1]

(iv) Give the 'file mode' available in the programming language which will be used to address this issue.

.....[1]

(c) The complete data file is created with the structure shown.

```
00001▼007
00002▼001
00003▼002
00004▼003
00005▼101
00006▼004
00007▼004

∫

00024▼002
00025▼003
00026▼007
00027▼007
00028▼102

∫

99867▼013
```

A new program is to be written to search the file.

The program will:

- input a bank code
- output a list of all the dispensers which belong to this bank
- output the total number of dispensers for this bank

An example of a run of the program is shown:

```
Enter bank code 007
00001
00011
00022
00026
00027

There are 5 dispensers for this bank
```

Write the **program code**. Do not attempt to include any validation checks.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[10]

5 A firm employs workers who assemble amplifiers. Each member of staff works an agreed number of hours each day.

The firm records the number of completed amplifiers made by each employee each day.

Management monitor the performance of all its workers.

Production data was collected for 3 workers over 4 days.

	Daily hours worked
Worker 1	5
Worker 2	10
Worker 3	10

Production data

	Worker 1	**Worker 2**	**Worker 3**
Day 1	10	20	9
Day 2	11	16	11
Day 3	10	24	13
Day 4	14	20	17

A program is to be written to process the production data.

(a) The production data is to be stored in a 2-dimensional array `ProductionData`, declared as follows:

```
DECLARE ProductionData ARRAY[1:4, 1:3] : INTEGER
```

(i) Describe **two** features of an array.

1 .....

.....

2 .....

.....[2]

(ii) Give the value of `ProductionData[3, 2]`.

.....[1]

(iii) Describe the information produced by the expression:

```
ProductionData[2, 1] + ProductionData[2, 2] + ProductionData[2, 3]
```

.....

.....[2]

(b) Complete the trace table for the pseudocode algorithm below.

```
FOR WorkerNum ← 1 TO 3
   WorkerTotal[WorkerNum] ← 0
ENDFOR

FOR WorkerNum ← 1 TO 3
   FOR DayNum ← 1 TO 4
      WorkerTotal[WorkerNum] ← WorkerTotal[WorkerNum] +
                                  ProductionData[DayNum, WorkerNum]
   ENDFOR
ENDFOR

FOR WorkerNum ← 1 TO 3
   WorkerAverage ← WorkerTotal[WorkerNum]/
                                   (4 * DailyHoursWorked[WorkerNum])
   IF WorkerAverage < 2
      THEN
         OUTPUT "Investigate", WorkerNum
   ENDIF
ENDFOR
```

WorkerNum	DayNum	WorkerAverage	OUTPUT

WorkerTotal		
1	2	3

[8]

(c) An experienced programmer suggests that the pseudocode would be best implemented as a procedure `AnalyseProductionData`.

Assume that both arrays, `DailyHoursWorked` and `ProductionData`, are available to the procedure from the main program and they are of the appropriate size.

```
PROCEDURE AnalyseProductionData(NumDays : INTEGER, NumWorkers : INTEGER)

   DECLARE ..............................................................................

   DECLARE ..............................................................................

   DECLARE ..............................................................................

   DECLARE ..............................................................................

   FOR WorkerNum ← 1 TO 3
      WorkerTotal[WorkerNum] ← 0
   ENDFOR

   FOR WorkerNum ← 1 TO 3
      FOR DayNum ← 1 TO 4
         WorkerTotal[WorkerNum] ← WorkerTotal[WorkerNum] +
                                     ProductionData[DayNum, WorkerNum]
      ENDFOR
   ENDFOR

   FOR WorkerNum ← 1 TO 3
      WorkerAverage ← WorkerTotal[WorkerNum]/
                                  (4 * DailyHoursWorked [WorkerNum])
      IF WorkerAverage < 2
         THEN
            OUTPUT "Investigate", WorkerNum
      ENDIF
   ENDFOR

ENDPROCEDURE
```

(i) Complete the declaration statements showing the local variables. [4]

(ii) The original pseudocode has been ‘pasted’ under the procedure header.

Circle all the places in the original pseudocode where changes will need to be made.

Write the changes which need to be made next to each circle. [3]

(iii) Write the statement for a procedure call which processes data for 7 days for 13 workers.

.....[1]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2015

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2022**

INSERT **2 hours**

6785454555–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (RW) 303746/2
© UCLES 2022

Functions

Note: An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

`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"`
`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 string `"BCD"`
`LENGTH(ThisString : STRING) RETURNS INTEGER` returns the integer value representing the length of string `ThisString` Example: `LENGTH("Happy Days")` returns `10`
`LCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the lower-case equivalent of `ThisChar` Alphabetic characters that are not upper case are returned unchanged. Example: `LCASE('W')` returns `'w'`
`UCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the upper-case equivalent of `ThisChar` Alphabetic characters that are not lower case are returned unchanged. Example: `UCASE('a')` returns `'A'`
`TO_UPPER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to upper case. Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`
`TO_LOWER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to lower case. Example: `TO_LOWER("JIM 803")` returns `"jim 803"`
`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>` returns a string representation of a numeric value. Note: `<datatype1>` may be `REAL` or `INTEGER`, <datatype2> may be `CHAR` or `STRING` Example: `NUM_TO_STR(87.5)` returns `"87.5"`
`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>` returns a numeric representation of a string. Note: `<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER` Example: `STR_TO_NUM("23.45")` returns `23.45`

String and Character Functions

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN` returns the value `TRUE` if `ThisString` represents a valid numeric value. Note: `<datatype>` may be `CHAR` or `STRING` Example: `IS_NUM("-12.36")` returns `TRUE`
`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`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`

Date Functions

Note: Date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2, etc. Example: `DAYINDEX(12/05/2020)` returns `3`
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a variable of type `DATE` with the value of `<Day>/<Month>/<Year>`
`TODAY() RETURNS DATE` returns a variable of type `DATE` with the value set to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: The function will generate an error if the file is not already open in `READ` mode.

Operators

Note: An error will be generated if an operator is used with a value or values of an incorrect type.

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: May also be used to concatenate a `CHAR` with a `STRING`
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	Used to compare two items of the same type. Returns `TRUE` if the condition is true, otherwise returns `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records.
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `count = 4` evaluates to `TRUE` when variable `count` contains the value `4`

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2023**

INSERT **2 hours**

0113520592-I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (LK) 315867/1
© UCLES 2023

An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and character functions

- A string of length 1 may be considered to be either of type `CHAR` or `STRING`
- A `CHAR` may be assigned to, or concatenated with, a `STRING`
- A `STRING` of length greater than 1 cannot be assigned to a `CHAR`

```
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"`

```
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`

```
TO_UPPER(x : <datatype>) RETURNS <datatype>
```

`<datatype>` may be `CHAR` or `STRING`

returns an object of type `<datatype>` formed by converting all characters of `x` to upper case.

Examples:

- `TO_UPPER("Error 803")` returns `"ERROR 803"`
- `TO_UPPER('a')` returns `'A'`

```
TO_LOWER(x : <datatype>) RETURNS <datatype>
```

`<datatype>` may be `CHAR` or `STRING`

returns an object of type `<datatype>` formed by converting all characters of `x` to lower case.

Examples:

- `TO_LOWER("JIM 803")` returns `"jim 803"`
- `TO_LOWER('W')` returns `'w'`

```
NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>
```

returns a string representation of a numeric value.
`<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING`
Example: `NUM_TO_STR(87.5)` returns `"87.5"`
If `x` is a negative value, the returned value will be a string beginning with the '–' character.

```
STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>
```

returns a numeric representation of a string.
`<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER`
Example: `STR_TO_NUM("23.45")` returns `23.45`

© UCLES 2023

```
IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN
```

returns `TRUE` if `ThisString` represents a valid numeric value.
`<datatype>` may be `CHAR` or `STRING`
Example: `IS_NUM("-12.36")` returns `TRUE`

```
ASC(ThisChar : CHAR) RETURNS INTEGER
```

returns an integer value (the ASCII value) of `ThisChar`
Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.

```
CHR(x : INTEGER) RETURNS CHAR
```

returns the character whose integer value (the ASCII value) is `x`
Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric functions

```
INT(x : REAL) RETURNS INTEGER
```

returns the integer part of `x`
Example: `INT(27.5415)` returns `27`

```
RAND(x : INTEGER) RETURNS REAL
```

returns a real number in the range `0` to `x` (**not** inclusive of `x`).
Example: `RAND(87)` could return `35.430729`

Date functions
Date format is assumed to be DD/MM/YYYY unless otherwise stated

```
DAY(ThisDate : DATE) RETURNS INTEGER
```

returns the current day number from `ThisDate`
Example: `DAY(04/10/2003)` returns `4`

```
MONTH(ThisDate : DATE) RETURNS INTEGER
```

returns the current month number from `ThisDate`
Example: `MONTH(04/10/2003)` returns `10`

```
YEAR(ThisDate : DATE) RETURNS INTEGER
```

returns the current year number from `ThisDate`
Example: `YEAR(04/10/2003)` returns `2003`

```
DAYINDEX(ThisDate : DATE) RETURNS INTEGER
```

returns the day index number from `ThisDate` where Sunday = 1, Monday = 2 etc.
Example: `DAYINDEX(07/11/2023)` returns 3

```
SETDATE(Day, Month, Year : INTEGER) RETURNS DATE
```

returns a value of type `DATE` with the value of `<Day>/<Month>/<Year>`
Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26/10/2003

```
TODAY() RETURNS DATE
```

returns a value of type `DATE` corresponding to the current date.

Text file functions

```
EOF(FileName : STRING) RETURNS BOOLEAN
```

returns `TRUE` if there are no more lines to be read from file `FileName`
The function will generate an error if the file is not already open in `READ` mode.

Operators

An error will be generated if an operator is used with a value(s) of an incorrect type.

`&`	concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values. Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values. Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value. Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another. Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another. Example `10 DIV 3` evaluates to `3`

Comparison operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	used to compare two items of the same type.
	evaluates to `TRUE` if the condition is true, otherwise evaluates to `FALSE`
	
	Notes:
	
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` and / or `STRING`
	• cannot be used to compare two records
	
	Examples:
	
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when `Count` contains the value 4

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2021**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **16** pages. Any blank pages are indicated.

DC (MB/FC) 200588/5
© UCLES 2021

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** **(i)** Complete the following table by giving the appropriate data type in each case.

Variable	Example data value	Data type
`Name`	`"Catherine"`	
`Index`	`100`	
`Modified`	`FALSE`	
`Holiday`	`25/12/2020`	

[4]

(ii) Evaluate each expression in the following table by using the initial data values shown in **part (a)(i)**.

Expression	Evaluates to
`Modified OR Index > 100`	
`LENGTH("Student: " & Name)`	
`INT(Index + 2.9)`	
`MID(Name, 1, 3)`	

[4]

(b) Each pseudocode statement in the following table contains an example of selection, assignment or iteration.

Put **one** tick ('✓') in the appropriate column for each statement.

Statement	Selection	Assignment	Iteration
`Index ← Index + 1`			
`IF Modified = TRUE THEN`			
`ENDWHILE`			

[3]

2 **(a)** Examine the following state-transition diagram.

(i) Complete the table with reference to the diagram.

	Answer
The number of transitions that result in a different state	
The number of transitions with associated outputs	
The label that should replace 'X'	
The final or halting state	

[4]

(ii) The current state is S1. The following inputs occur.

1. Low level detected
2. Low level detected
3. Low level detected
4. Low level detected

Give the number of outputs and the current state.

Number of outputs .....

Current state .....

[2]

(b) A system is being developed to help manage book loans in a library.

Registered users may borrow books from the library for a period of time.

(i) State **three** items of data that must be stored for each loan.

1 .....

2 .....

3 .....
[2]

(ii) State **one** item of data that will be required in the library system but does not need to be stored for each loan.

..... [1]

(iii) One operation that manipulates the data stored for each loan, would produce a list of all overdue books.

Identify **two other** operations.

Operation 1 .....

.....

Operation 2 .....

.....
[2]

3 The following diagram represents an Abstract Data Type (ADT).

(a) Identify this type of ADT.

..... [1]

(b) Give the technical term for the item labelled **A** in the diagram.

..... [1]

(c) Give the technical term for the item labelled **B** in the diagram.

Explain the meaning of the value given to this item.

Term .....

Meaning .....

.....

.....
[2]

(d) Complete the diagram to show the ADT after the data has been sorted in alphabetical order.

[2]

4 A teacher uses a paper-based system to store marks for a class test. The teacher requires a program to assign grades based on these results.

The program will output the grades together with the average mark.

Write a detailed description of the algorithm that will be needed.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

5 **(a)** A student is learning about arrays.

She wants to write a program to:

- declare a 1D array `RNum` of 100 elements of type `INTEGER`
- assign each element a random value in the range 1 to 200 inclusive
- count and output how many numbers generated were between 66 and 173 inclusive.

(i) Write pseudocode to represent the algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(ii) The student decides to modify the algorithm so that each element of the array will contain a unique value.

Describe the changes that the student needs to make to the algorithm.

.....

.....

.....

.....

.....

..... [3]

(b) The following is a pseudocode function.

Line numbers are given for reference only.

```
01  FUNCTION StringClean(InString : STRING) RETURNS STRING
02
03     DECLARE NextChar : CHAR
04     DECLARE OutString : STRING
05     DECLARE Counter : INTEGER
06
07     OutString ← ""
08
09     FOR Counter ← 1 TO LENGTH(InString)
10        NextChar ← MID(InString, Counter, 1)
11        NextChar ← LCASE(NextChar)
12        IF NOT((NextChar < 'a') OR (NextChar > 'z')) THEN
13           OutString ← OutString & NextChar
14        ENDIF
15     NEXT Counter
16
17     RETURN OutString
18
19  ENDFUNCTION
```

(i) Examine the pseudocode and complete the following table.

	Answer
Give a line number containing an example of an initialisation statement.	
Give a line number containing the start of a repeating block of code.	
Give a line number containing a logic operation.	
Give the number of parameters to the function `MID()`.	

[4]

(ii) Write a simplified version of the statement in line `12`.

.....

..... [2]

BLANK PAGE

6 A procedure `CountVowels()` will:

- be called with a string containing alphanumeric characters as its parameter
- count and output the number of occurrences of each vowel (a, e, i, o, u) in the string
- count and output the number of occurrences of the other alphabetic characters (as a single total).

The string may contain both upper and lower case characters.

Each count value will be stored in a unique element of a global 1D array `CharCount` of type `INTEGER`. The array will contain six elements.

Write pseudocode for the procedure `CountVowels()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

7 A procedure, `FormatName()`:

- is called with a string containing words and spaces as its parameter. In this context, a word is any sequence of characters that does not contain a space character.
- creates a new formatted string from this string with the following requirements:
    1. Any leading spaces removed (spaces before the first word).
    2. Any trailing spaces removed (spaces after the last word).
    3. Any multiple spaces between words converted to a single space.
    4. All characters converted to lower case.

The `FormatName()` procedure has been written in a programming language and is to be tested using the black-box method.

(a) Give a test string that could be used to show that all **four** formatting requirements have been applied correctly.

Use the symbol '∇' to represent a space character.

..... [3]

(b) The `FormatName()` procedure should assign a value to the global variable `FString`.

There is a fault in the program, which means that the assignment does not always take place.

Explain **two** ways of exposing the fault.

.....

.....

.....

.....

..... [2]

8 A program is needed to take a string containing a full name and to produce a new string of initials.

Some words in the full name will be ignored. For example, “the”, “and”, “of”, “for” and “to” may all be ignored.

Each letter of the new string must be upper case.

For example:

Full name	Initials
Integrated Development Environment	IDE
The American Standard Code for Information Interchange	ASCII

The programmer has decided to use the following global variables:

- a ten element 1D array `IgnoreList` of type `STRING` to store the ignored words
- a string `FNString` to store the full name string.

Assume that:

- each alphabetic character in the full name string may be either upper or lower case
- the full name string contains at least one word.

The programmer has started to define program modules as follows:

Module	Description
`GetStart()`	• Called with an `INTEGER` as its parameter, representing the number of a word in `FNString` • Returns the character start position of that word in `FNString` or returns `-1` if that word does not exist • For example: `GetStart(3)` applied to `"hot and cold"` returns `9`
`GetWord()`	• Called with the position of the first character of a word in `FNString` as its parameter • Returns the word from `FNString` • For example: if `FNString` contains the string `"hot and cold"`, `GetWord(9)` returns `"cold"`
`IgnoreWord()`	• Called with a `STRING` parameter representing a word • Searches for the word in the `IgnoreList` array • Returns `TRUE` if the word is found, otherwise returns `FALSE`
`GetInitials()`	• Processes the sequence of words in the full name one word at a time • Calls `GetStart()`, `GetWord()` and `IgnoreWord()` to process each word to form the new string • Outputs the new string

(a) Write pseudocode for the module `IgnoreWord()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) Write pseudocode for the module `GetInitials()`.

..... [8]

BLANK PAGE

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2022**

INSERT **2 hours**

5334515327–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (SLM) 320146
© UCLES 2022

Functions

Note: an error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

`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"`
`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`
`LCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the lower-case equivalent of `ThisChar` Non upper-case alphabetic characters are returned unchanged. Example: `LCASE('W')` returns `'w'`
`UCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the upper-case equivalent of `ThisChar` Non lower-case alphabetic characters are returned unchanged. Example: `UCASE('a')` returns `'A'`
`TO_UPPER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to upper case. Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`
`TO_LOWER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to lower case. Example: `TO_LOWER("JIM 803")` returns `"jim 803"`
`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>` returns a string representation of a numeric value. Note: `<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING` Example: `NUM_TO_STR(87.5)` returns `"87.5"`
`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>` returns a numeric representation of a string. Note: `<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER` Example: `STR_TO_NUM("23.45")` returns `23.45`

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN` returns the value `TRUE` if `ThisString` represents a valid numeric value. Note: `<datatype>` may be `CHAR` or `STRING` Example: `IS_NUM("-12.36")` returns `TRUE`
`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66` etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'` etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`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`

Date Functions

Note: date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2 etc. Example: `DAYINDEX(12/05/2020)` returns `3`
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a variable of type `DATE` with the value of `<Day>/<Month>/<Year>` Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26 October 2003
`TODAY() RETURNS DATE` returns a variable of type `DATE` corresponding to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: the function will generate an error if the file is not already open in `READ` mode.

Operators

Note: an error will be generated if an operator is used with a value or values of an incorrect type.

`&`	concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
 	
`>`
 	
`<`
 	
`>=`
 	
`<=`
 	
`<>`
	used to compare two items of the same type
	returns `TRUE` if the condition is true, otherwise returns `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when variable `Count` contains the value `4`

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

4420502994

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2018**

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 **16** printed pages.

DC (NH/CT) 151783/3
© UCLES 2018

CAMBRIDGE International Examinations

Question 1 begins on the next 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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2018

1 **(a)** The following table contains statements written in pseudocode.

Show the type of programming construct each statement represents.

Put a tick (✓) in the appropriate column for each statement.

Statement	Assignment	Selection	Repetition (Iteration)
`CASE OF TempSensor1`			
`ELSE`			
`REPEAT`			
`ENDFOR`			
`DayNumber ← DayNumber + 1`			
`Error ← TRUE`			

[6]

(b) **(i)** The following table contains statements written in pseudocode.

Give the most appropriate data type for the variable used in each statement.

Statement	Data type
`Revision ← 500`	
`FuelType ← 'P'`	
`MinValue ← -6.3`	
`ServiceDue ← FALSE`	
`ModelRef ← "W212DEC15"`	

[5]

(ii) The following table contains statements written in pseudocode.

Complete the table by evaluating each expression using the values from part **(b)(i)**.

If any expression is invalid, write “ERROR” in the **Evaluates to** column.

For the built-in functions list, refer to the **Appendix** on page 16.

Expression	Evaluates to
`"Month: " & MID(ModelRef, 5, 3)`	
`INT(MinValue * 2)`	
`ASC(Revision)`	
`Revision > 500`	
`ServiceDue = TRUE OR FuelType = 'P'`	

[5]

2 Shop customers have a discount card with a unique card number. Customers collect points when they buy items. At the end of each year, customers are given bonus (extra) points related to the total amount they have spent during the year, and the number of points they have on their card.

The function `CalcBonus()` takes the card number as a parameter. It returns the bonus points for the customer. A flowchart for the function is shown.

The function uses the following variables and functions.

Identifier	Data type	Description
`CardNum`	`STRING`	A numeric string representing the unique card number
`Points`	`INTEGER`	The number of points collected
`Spend`	`REAL`	The total amount that customer has spent during the year
`Bonus`	`INTEGER`	The number of bonus points
`GetPoints()`	`FUNCTION`	Takes the card number as a parameter and returns the number of points already collected
`GetSpend()`	`FUNCTION`	Takes the card number as a parameter and returns the total amount that customer has spent during the year

(a) **(i)** Write **pseudocode** for the `CalcBonus()` function.

Your solution should follow the flowchart for the function as closely as possible.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

(ii) The function `GetCardNumber()` prompts the user to input a card number until the number input is valid.

A valid card number has 16 characters. Each character is a numeric character (`'0'` to `'9'`).

Write **pseudocode** to complete the `GetCardNumber()` function.

You should refer to the function `IS_NUM()` in the **Appendix** on page 16.

```
FUNCTION GetCardNumber() RETURNS STRING
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
ENDFUNCTION
```

[6]

(b) The function `CalcBonus()` is written in a high-level language.

(i) The function is tested using black-box testing and does not contain any syntax errors.

Name **and** describe **one** other type of error that black-box testing could find.

Name .....

Description .....

.....

.....
[2]

(ii) The function `CalcBonus()` is tested using white-box testing.

State **two** different pairs of values for `Spend` and `Points` that can be used to test different paths through the function. Justify your choices.

`Spend` ..... `Points` .....

Justification .....

.....

.....

`Spend` ..... `Points` .....

Justification .....

.....

.....
[4]

(c) Name **two** types of program maintenance **and** state the reason why each is needed.

Name .....

Reason .....

.....

.....

Name .....

Reason .....

.....

.....
[4]

3 An array contains 100 integer values. An algorithm will find the maximum and minimum values stored in the array.

(a) A programmer has started to write this program using a conditional loop.

Name a more appropriate loop structure for this task **and** justify your choice.

Name .....

Justification .....

.....

.....
[2]

(b) Outline the steps the program will need to follow to implement the algorithm.

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

.....

.....

.....

.....

.....

.....

.....

.....[3]

Question 4 begins on the next page.

4 Part of a program written in pseudocode is shown.

```
010 DECLARE ThisArray : ARRAY [1:100, 1:10] OF STRING
011 DECLARE LastElement : INTEGER
…
100 FUNCTION Update(NewData : STRING) RETURNS INTEGER
101    DECLARE ArrayIndex : INTEGER
102    DECLARE Found : BOOLEAN
103
104    ArrayIndex ← 1
105    Found ← FALSE
106
107    WHILE ArrayIndex <= LastElement AND Found = FALSE
108       IF ThisArray[ArrayIndex, 1] > NewData
109         THEN
110           Found ← TRUE
111         ELSE
112           ArrayIndex ← ArrayIndex + 1
113       ENDIF
114    ENDWHILE
115
116    IF Found = TRUE
117      THEN
118         CALL Insert(ArrayIndex, NewData)
119      ELSE
120         ArrayIndex ← 0
121    ENDIF
122
123    RETURN ArrayIndex
124 ENDFUNCTION
```

(a) **(i)** Examine the pseudocode **and** complete the following table.

	Answer
The name of a global identifier	
The name of a user-defined procedure	
The scope of `ArrayIndex`	
The number of dimensions of `ThisArray`	
The scope of `NewData`	

[5]

(ii) Describe in detail, the purpose of lines `107` to `114` in the `Update()` function.
Do **not** use pseudocode in your answer.

.....

.....

.....

.....

.....

.....[4]

(b) Line `118` of the function `Update()` calls the subroutine `Insert()`. A designer decides to convert `Insert()` from a procedure to a function. `Insert()` returns `TRUE` or `FALSE`.

A programmer must amend the function `Update()` as follows:

- If `Insert()` returns `FALSE` then the function `Update()` returns `-1`
- If `Insert()` returns `TRUE` then the function `Update()` returns the value as before.

Write **program code** to implement the **amended** `Update()` function.

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]

(c) The function `Update()` is an example of a module within a program.

Describe the mechanism that supports the transfer of values between modules.

.....

.....

.....

.....

.....[2]

(d) **(i)** `CharArray` is a 1D array of type `CHAR`. It contains 200 elements.

Write **program code** to change all the numeric characters (`'0'` to `'9'`) in `CharArray` to `'*'`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....[3]

(ii) A programmer decides to declare `LastElement` as a constant instead of a variable.

Write a statement in **pseudocode** to declare `LastElement` as the value 200.

.....[1]

Question 5 begins on the next page.

5 The function `ReadFileLine()` returns a specific line from a text file.

The function takes two parameters:

Identifier	Data type	Description
`FileName`	`STRING`	The name of the text file
`FileLine`	`INTEGER`	The line number that is required

The following pseudocode gives an example of the use of the function.

```
FileData ← ReadFileLine(FileName, FileLine)
```

The function `ReadFileLine()` will:

- open the file, `FileName`
- read each line from the file until line `FileLine` is found or the end of the file is reached
- if the line exists, return the string from this line; otherwise return the string `"****"`

Write **pseudocode** for the `ReadFileLine()` function.

.....[10]

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 string `"BCD"`

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

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

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

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

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

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

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

Example: `INT(27.5415)` returns `27`

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

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

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

Example: `IS_NUM("1234X67")` 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`

© UCLES 2018

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

9702757866

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2015**

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 **19** printed pages and **1** blank page.

DC (NH/CGW) 95372/4
© UCLES 2015

CAMBRIDGE International Examinations

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

1 Computer programs have to evaluate expressions.

Study the sequence of pseudocode statements.

Write down the value assigned to each variable.

```
DECLARE h, w, r, Perimeter, Area : REAL
DECLARE A, B, C, D, E            : BOOLEAN
```

```
h ← 13.6
w ← 6.4
Perimeter ← (h + w) * 2
```

(i) Perimeter ..... [1]

```
r ← 10
Area 3.142 * r^2
```

(ii) Area ..... [1]

```
Z ← 11 + r / 5 + 3
```

(iii) Z ..... [1]

```
A ← NOT(r > 10)
```

(iv) A ..... [1]

2 A programmer uses an Integrated Development Environment (IDE) for all program development.

(i) Describe what is meant by an IDE.

.....

.....

.....

.....[2]

(ii) Name **three** features you would expect to be available in an IDE to help initial error detection or debugging.

1 .....

.....

2 .....

.....

3 .....

.....[3]

3 A program is to simulate the operation of a particular type of logic gate.

- The gate has two inputs (TRUE or FALSE) which are entered by the user.
- The program will display the output (TRUE or FALSE) from the gate.

The program uses the following identifiers in the pseudocode below:

Identifier	Data type	Description
`InA`	`BOOLEAN`	Input signal
`InB`	`BOOLEAN`	Input signal
`OutZ`	`BOOLEAN`	Output signal

```
INPUT InA
INPUT InB
IF (InA = FALSE AND InB = FALSE) OR (InA = FALSE AND InB = TRUE)
                                    OR (InA = TRUE AND InB = FALSE)
   THEN
      OutZ ← TRUE
   ELSE
      OutZ ← FALSE
ENDIF
OUTPUT OutZ
```

(a) The programmer chooses the following four test cases.

Show the output (`OutZ`) expected for each test case.

Test case	Input		Output `OutZ`
	`InA`	`InB`	
1	`TRUE`	`TRUE`	
2	`TRUE`	`FALSE`	
3	`FALSE`	`TRUE`	
4	`FALSE`	`FALSE`	

[4]

(b) The selection statement (lines `03` – `08`) could have been written with more simplified logic.

Rewrite this section of the algorithm in **pseudocode**.

.....

.....

.....

.....

.....

.....[3]

4 A program is to be written to calculate the discount given on purchases.

A purchase may qualify for a discount depending on the amount spent. The purchase price (`Purchase`), the discount rate (`DiscountRate`) and amount paid (`Paid`) is calculated as shown in the following pseudocode algorithm.

```
INPUT Purchase

IF Purchase > 1000
   THEN
      DiscountRate ← 0.10
   ELSE
      IF Purchase > 500
         THEN
            DiscountRate ← 0.05
         ELSE
            DiscountRate ← 0
      ENDIF
ENDIF

Paid ← Purchase * (1 - DiscountRate)
OUTPUT Paid
```

The algorithm is also to be documented with a program flowchart.

Complete the flowchart by:

- filling in the flowchart boxes
- labelling, where appropriate, lines of the flowchart

[6]

5 A driver buys a new car.

The value of the car reduces each year by a percentage of its current value.

The percentage reduction is:

- in the first year, 40%
- in each following year, 20%

The driver writes a program to predict the value of the car in future years.

The program requirements are:

- enter the cost of the new car (to nearest $)
- calculate and output the value of the car at the end of each year
- the program will end when either the car is nine years old, or when the value is less than $1000

(a) Study the incomplete pseudocode which follows in **part (b)** and fill in the identifier table.

Identifier	Data type	Description

[3]

(b) Complete the pseudocode for this design.

```
OUTPUT "Enter purchase price"
INPUT PurchasePrice

CurrentValue ← ..............................................................
YearCount ← 1

WHILE ..................................... AND ..............................

   IF ..........................................................................
      THEN
         CurrentValue ← CurrentValue * (1 - 40 / 100)
      ELSE
         CurrentValue ← ...........................................................
   ENDIF

   OUTPUT YearCount, CurrentValue

   ..............................................................................
ENDWHILE
```

[6]

6 A firm employs five staff who take part in a training programme. Each member of staff must complete a set of twelve tasks which can be taken in any order. When a member of staff successfully completes a task, this is recorded.

A program is to be produced to record the completion of tasks for the five members of staff.

To test the code, the programmer makes the program generate test data.

The program generates pairs of random numbers:

- the first, in the range, 1 to 5 to represent the member of staff
- the second, in the range, 1 to 12 to represent the task

Each pair of numbers simulates the completion of one task by one member of staff.

(a) Explain why the generation of 60 (5 staff x 12 tasks) pairs of random numbers will not simulate all tasks completed by all staff.

.....

.....

.....[2]

(b) Data is currently recorded manually as shown.

Staff number	**Task number**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1												
2												
3				✓								
4												
5								✓				

The table shows that two members of staff have each successfully completed one task.

The program must use a suitable data structure to store, for all staff:

- tasks successfully completed
- tasks not yet successfully completed

The program will output the staff number and task number in the order in which tasks are completed.

The program design in pseudocode is produced as follows:

```
DECLARE StaffNum          : INTEGER
DECLARE TaskNum           : INTEGER
DECLARE ..................................................................................
DECLARE NewStaffTask      : BOOLEAN

CALL InitialiseTaskGrid
Completed ← 0
WHILE Completed <> 60
   NewStaffTask ← FALSE
   WHILE NewStaffTask = FALSE
      StaffNum ← RANDOM(1,5)       //generates a random number
      TaskNum ← RANDOM(1,12)       //in the given range
      IF TaskGrid[StaffNum, TaskNum] = FALSE
         THEN
            TaskGrid[StaffNum, TaskNum] ← TRUE
            NewStaffTask ← TRUE
            OUTPUT StaffNum, TaskNum
      ENDIF
   ENDWHILE
   Completed ← Completed + 1
ENDWHILE
OUTPUT "Staff Task Count", Completed

// end of main program

PROCEDURE InitialiseTaskGrid()
   DECLARE i : INTEGER
   DECLARE j : INTEGER
   FOR i ← 1 TO 5
      FOR j ← 1 TO 12
         TaskGrid[i, j] ← FALSE
      ENDFOR
   ENDFOR
ENDPROCEDURE
```

Study the pseudocode and answer the questions below.

Give the line number for:

(i) The declaration of a `BOOLEAN` global variable. ..... [1]

(ii) The declaration of a local variable. ..... [1]

(iii) The incrementing of a variable used as a counter, but not to control a 'count controlled' loop.

..... [1]

(iv) A statement which uses a built-in function of the programming language.

..... [1]

(c) (i) State the number of parameters of the `InitialiseTaskGrid` procedure.

..... [1]

(ii) Copy the condition which is used to control a 'pre-condition' loop.

.....[1]

(iii) Explain the purpose of lines `13` – `18`.

.....

.....

.....

.....

.....

.....[3]

(iv) Give the global variable that needs to be declared at line `03`.

.....[2]

(d) Line `17` in the pseudocode outputs the staff number and the task number.

A new requirement is to display the name of the member of staff given in the table.

Write a CASE structure using variable `StaffNum`.

Assign to a new variable `StaffName` the appropriate staff name.

Staff number	Staff name
1	Sadiq
2	Smith
3	Ho
4	Azmah
5	Papadopoulos

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

Question 7 begins on page 14.

7 ASCII character codes are used to represent a single character.

Part of the code table is shown below.

ASCII code table (part)

Character	Decimal	Character	Decimal	Character	Decimal
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

Some pseudocode statements follow which use the built-in functions below:

`ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR`
returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.
For example: `ONECHAR("Barcelona", 3)` returns 'r'.

`CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER`
returns the number of characters in the string `ThisString`.
For example: `CHARACTERCOUNT("BRAZIL")` returns 6.

`CHR(ThisInteger : INTEGER) RETURNS CHAR`
returns the character with ASCII code `ThisInteger`.
For example: `CHR(65)` returns character 'A'.

`ASC(ThisCharacter : CHAR) RETURNS INTEGER`
returns the ASCII value for character `ThisCharacter`.
For example: `ASC('A')` returns 65.

(a) Show the values stored by variables `A`, `B`, `C` and `D`.

The & operator is used to concatenate two strings.

```
Num1 ← 15
A ← CHR(67) & CHR(65) & CHR(84)
B ← ASC('P') - ASC('F') + 3
C ← ASC(ONECHAR("BISCUITS", 3))
D ← CHARACTERCOUNT("New York City") + 2
```

(i) `A` ..... [1]

(ii) `B` ..... [1]

(iii) `C` ..... [1]

(iv) `D` ..... [1]

(b) A program is to be written which accepts a string and then calculates a numeric value from this string. The input string and the calculated value are then to be sent to a remote computer over a communications link.

Study the following pseudocode:

```
OUTPUT "Enter string"
INPUT MyString
StringTotal ← 0

FOR i ← 1 TO CHARACTERCOUNT(MyString)
   NextNum ← ASC(ONECHAR(MyString, i))
   StringTotal ← StringTotal + NextNum
ENDFOR

OUTPUT MyString, StringTotal
```

Write the above pseudocode algorithm as **program code**.

There is no need to show the declaration of variables or comment statements.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

(c) Explain the purpose of sending the value of `StringTotal` to the remote computer, in addition to `MyString`.

.....

.....

.....

.....[2]

8 In this question you will need to use the given pseudocode built-in function:

```
ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
```
returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.
For example: `ONECHAR("Barcelona", 3)` returns 'r'.

(a) Give the value assigned to variable `y` by the following statement:

`y ← ONECHAR("San Francisco", 6)` `y` ..... [1]

A program reads a string entered by the user. The string represents the addition or subtraction of two fractions. Each part of the fraction within the string is always a single digit only and the top digit is always less than the bottom digit.

Example strings are: `"3/8+3/5"` and `"5/8-1/4"`

The program steps are:
- the user enters the string
- the program isolates each digit and the operator
- the program computes the answer as either:
  - a fraction
  - a whole number followed by a fraction
  - a whole number
- the program displays the answer to the user

The identifier table shows the variables to be used to store the characters in the string as shown in the diagram.

Identifier	Data type	Description
`FractionString`	`STRING`	String input by user. For example: `"5/8-1/4"`
`N1Char`	`CHAR`	See diagram
`N2Char`	`CHAR`	See diagram
`N3Char`	`CHAR`	See diagram
`N4Char`	`CHAR`	See diagram
`Op`	`CHAR`	See diagram

(b) Study the sequence of pseudocode statements.

Show the values assigned to each variable.

```
FractionString ← "3/7+2/9"
N3Char ← ONECHAR(FractionString, 5)
Op ← ONECHAR(FractionString, 4)
```

(i) `N3Char` ..... [1]

(ii) `Op` ..... [1]

(iii) Complete the function call to isolate the character `'9'` from `FractionString`.

```
FractionString ← "3/7+2/9"
ONECHAR(FractionString, ..............)
```

[1]

The following additional variables are to be used by the program:

Identifier	Data type	Description
N1	INTEGER	The number value of N1Char
N2	INTEGER	The number value of N2Char
N3	INTEGER	The number value of N3Char
N4	INTEGER	The number value of N4Char
TopAnswer	INTEGER	The numerator of the fraction answer
BottomAnswer	INTEGER	The denominator of the fraction answer

(c) The following pseudocode uses these additional built-in functions:

`TONUM(ThisDigit : CHAR) RETURNS INTEGER` returns the integer value of character `ThisDigit` For example: `TONUM('8')` returns digit `8`.

`TOSTR(ThisNumber : INTEGER) RETURNS STRING` returns the string value of integer `ThisNumber` For example: `TOSTR(27)` returns `"27"`.

Study the pseudocode.

Complete the **three** dry runs for the three given values of `FractionString`.

```
OUTPUT "Enter the expression"
INPUT FractionString

// isolate each number digit and assign its number value
N1Char ← ONECHAR(FractionString, 1)
N1 ← TONUM(N1Char)
N2Char ← ONECHAR(FractionString, 3)
N2 ← TONUM(N2Char)
N3Char ← ONECHAR(FractionString, 5)
N3 ← TONUM(N3Char)
N4Char ← ONECHAR(FractionString, 7)
N4 ← TONUM(N4Char)

BottomAnswer ← N2 * N4

Op ← ONECHAR(FractionString, 4)
IF Op = '+'
   THEN
      // add fractions
      TopAnswer ← (BottomAnswer/N2) * N1 + (BottomAnswer/N4) * N3
   ELSE
      // subtract fractions
      TopAnswer ← (BottomAnswer/N2) * N1 - (BottomAnswer/N4) * N3
ENDIF

IF TopAnswer = BottomAnswer
   THEN
      OUTPUT '1'
   ELSE
      IF TopAnswer > BottomAnswer
         THEN
            TopAnswer ← TopAnswer MOD BottomAnswer
            // the & operator joins strings or character values
            OUTPUT "1 " & TOSTR(TopAnswer) & "/" & TOSTR(BottomAnswer)
         ELSE
            OUTPUT TOSTR(TopAnswer) & "/" & TOSTR(BottomAnswer)
      ENDIF
ENDIF
```

(i) `FractionString ← "2/5-3/8"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[2]

(ii) `FractionString ← "3/4+1/4"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[2]

(iii) `FractionString ← "7/9+2/3"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[3]

(d) The programmer writes code from the given pseudocode design. The program works, but the design is limited.

The programmer is to make amendments to the design following suggested specification changes.

(i) State the name for this type of maintenance.

.....[1]

(ii) Describe **three** specification changes which will make the program more useful.

1 .....

.....

2 .....

.....

3 .....

.....[3]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2015

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2022**

INSERT **2 hours**

5904136477-I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (SLM) 320148
© UCLES 2022

Functions

Note: an error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

`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"`
`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`
`LCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the lower-case equivalent of `ThisChar` Non upper-case alphabetic characters are returned unchanged. Example: `LCASE('W')` returns `'w'`
`UCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the upper-case equivalent of `ThisChar` Non lower-case alphabetic characters are returned unchanged. Example: `UCASE('a')` returns `'A'`
`TO_UPPER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to upper case. Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`
`TO_LOWER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to lower case. Example: `TO_LOWER("JIM 803")` returns `"jim 803"`
`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>` returns a string representation of a numeric value. Note: `<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING` Example: `NUM_TO_STR(87.5)` returns `"87.5"`
`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>` returns a numeric representation of a string. Note: `<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER` Example: `STR_TO_NUM("23.45")` returns `23.45`

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN` returns the value `TRUE` if `ThisString` represents a valid numeric value. Note: `<datatype>` may be `CHAR` or `STRING` Example: `IS_NUM("-12.36")` returns `TRUE`
`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66` etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'` etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`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`

Date Functions

Note: date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2 etc. Example: `DAYINDEX(12/05/2020)` returns `3`
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a variable of type `DATE` with the value of `<Day>/<Month>/<Year>` Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26 October 2003
`TODAY() RETURNS DATE` returns a variable of type `DATE` corresponding to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: the function will generate an error if the file is not already open in `READ` mode.

Operators

Note: an error will be generated if an operator is used with a value or values of an incorrect type.

`&`	concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
 	
`>`
 	
`<`
 	
`>=`
 	
`<=`
 	
`<>`
	used to compare two items of the same type
	returns `TRUE` if the condition is true, otherwise returns `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when variable `Count` contains the value `4`

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

4253008298

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2017**

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 **14** printed pages and **2** blank pages.

DC (NF) 133468/3
© UCLES 2017

CAMBRIDGE International Examinations

1 **(a)** **(i)** Procedural high-level languages usually support different data types.

Give an appropriate data type for each data value in the following table:

Data value	Data type
`FALSE`	
`03/03/2013`	
`35`	
`"INTEGER"`	
`3.5`	
`"35"`	

[6]

(ii) The following is a declaration in a high-level language:

```
DEFINE MyGrade[1 to 100]
```

State the data structure of variable `MyGrade`.

..... [1]

(iii) An experienced programmer is presented with program code in an unfamiliar high-level language.

State **two** features of the code that the programmer should be able to recognise.

1 .....

.....

2 .....

.....

[2]

(b) **(i)** In the ASCII character set 'A' is represented by the value 65. The values representing the other characters of the alphabet follow in sequence, so 'B' is represented by 66, 'C' by 67 and so on.

The following table represents consecutive memory locations. Each memory location stores one byte.

Complete the table to show how the string "CAGE" may be stored in memory using the ASCII set.

Address	Data
100	
101	
102	
103	
104	
105	

[2]

(ii) In a high-level language, a `LENGTH` function is used to return the number of characters in a string.

Explain what is stored in addition to the string characters to allow this function to determine this number.

.....

.....

.....

..... [2]

(c) Functions and procedures are subroutines.

Explain why parameters are used with subroutines.

.....

.....

.....

.....

.....

..... [3]

(d) The following pseudocode is an example of a `CASE` structure.

```
CASE OF MyMark
   75 to 100: MyGrade ← "Distinction"
   35 to 74: MyGrade ← "Pass"
   0 to 34: MyGrade ← "Fail"
   OTHERWISE: OUTPUT "Invalid value entered"
ENDCASE
```

(i) Describe what will happen if the pseudocode is tested when `MyMark` has the following values:

27 .....

.....

101 .....

.....
[2]

(ii) Use **pseudocode** to write an `IF` statement with the same functionality.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

Question 2 begins on the next page.

2 A 1D array, `ClassName`, of type `STRING` contains 100 elements.

The following pseudocode represents a simple algorithm to process the array.

```
DECLARE SearchValue : STRING
DECLARE FoundFlag : BOOLEAN
DECLARE Index : INTEGER

INPUT SearchValue
FoundFlag ← FALSE
Index ← 1

WHILE Index < 101 AND FoundFlag = False
   IF ClassName[Index] = SearchValue
      THEN
         OUTPUT Index
         FoundFlag ← TRUE
   ENDIF
   Index ← Index + 1
ENDWHILE

IF FoundFlag = FALSE
   THEN
      OUTPUT "Not found"
ENDIF
```

(a) Describe the purpose of the algorithm.

.....

.....

.....

.....

.....

..... [2]

(b) Draw a program flowchart to represent this algorithm.

Note that variable declarations are not required in program flowcharts.

[9]

3 A 1D array, `Product`, of type `STRING` is used to store information about a range of products in a shop. There are 100 elements in the array. Each element stores one data item.

The format of each data item is as follows:

```
<ProductID><ProductName>
```

- `ProductID` is a four-character string of numerals
- `ProductName` is a variable-length string

The following pseudocode is an initial attempt at defining a procedure, `ArraySort`, which will perform a bubble sort on `Product`. The array is to be sorted in ascending order of `ProductID`. Line numbers have been added for identification purposes only.

```
01   PROCEDURE SortArray
02     DECLARE Temp : CHAR
03     DECLARE FirstID, SecondID : INTEGER
04     FOR I ← 1 TO 100
05       FOR J ← 2 TO 99
06          FirstID ← MODULUS(LEFT(Product[J], 6))
07          SecondID ← MODULUS(LEFT(Product[J + 1], 6))
08          IF FirstID > SecondID
09             THEN
10                Temp ← Product[I]
11                Product[I] ← Product[J + 1]
12                Product[J + 1] ← Temp
13        ENDFOR
14          ENDIF
15     ENDFOR
16   ENDPROCEDURE
```

The pseudocode on page 8 contains a number of errors. Complete the following table to show:

- the line number of the error
- the error itself
- the correction that is required.

Note:

- If the same error occurs on more than one line, you should only refer to it ONCE.
- Lack of optimisation should not be regarded as an error.

Line number	Error	Correction
01	Wrong procedure name – “SortArray”	`PROCEDURE ArraySort`

[8]

4 Programming languages provide built-in functions to generate random numbers.
To be truly random, the frequency of each number generated should be the same.

You are required to write program code to test the random number generator of your chosen language.

The test should:

- generate a given number of random numbers between 1 and 10 inclusive
- keep a count of the number of times each number is generated
- calculate the expected frequency of each number 1 to 10
- output the actual frequency of each number 1 to 10
- output the difference between the actual frequency and the expected frequency.

The program code should be written as a procedure. In pseudocode, the procedure heading will be:

```
PROCEDURE TestRandom(Repetitions AS INTEGER)
```

The parameter, `Repetitions`, contains a value representing the total number of random numbers that should be generated.

The following example shows the expected output for the procedure call, `TestRandom(200)`.

```
   The expected frequency is 20.

Number     Frequency     Difference
  1           17            -3
  2           21             1
  3           12            -8
  4           28             8
  5           20             0
  6           19            -1
  7           21             1
  8           16            -4
  9           24             4
  10          22             2
```

(a) Write **program code** for the procedure, `TestRandom`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [16]

(b) Name **three** features of a typical IDE that would help a programmer to debug a program.

Explain how each of these could be used in the debugging of the `TestRandom` procedure from **part (a)**.

Feature 1 .....

Explanation .....

.....

.....

.....

Feature 2 .....

Explanation .....

.....

.....

.....

Feature 3 .....

Explanation .....

.....

.....

.....

[6]

(c) The procedure is developed and run using the call `TestRandom(200)`. No system errors are produced.

To ensure that the procedure works correctly, you need to check the output.

Describe **two** checks you should make to suggest the program works correctly.

1 .....

.....

.....

2 .....

.....

.....

[2]

5 A sports club maintains a record of the email address of each of its members. The details are stored in a text file, `EmailDetails.txt`. The format of each line of the text file is as follows:

```
<MembershipNumber><EmailAddress>
```

- `MembershipNumber` is a four-character string of numerals
- `EmailAddress` is a variable-length string

When members leave the club their details need to be removed from the file.

A procedure, `RemoveDetails` is required. This will perform the following actions:

1. Input the `MembershipNumber` of the club member to be removed
2. Create a new file, `NewEmailDetails.txt`
3. Copy all the lines from `EmailDetails.txt` to `NewEmailDetails.txt`, except the line with the matching `MembershipNumber`

Write **pseudocode** for the procedure, `RemoveDetails`.

For the built-in functions list, refer to the **Appendix** on page 15.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [9]

BLANK PAGE

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

```
MODULUS(x : INTEGER, y : INTEGER) RETURNS INTEGER
```

returns the remainder when `x` is divided by `y` using integer arithmetic.
Example: `MODULUS(5, 2)` will return `1`

```
INT(x : REAL) RETURNS INTEGER
```

returns the integer part of `x`.
Example: `INT(27.5415)` returns `27`

```
LENGTH(ThisString : STRING) RETURNS INTEGER
```

returns the integer value representing the length of string `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 string `"ABC"`

```
RIGHT(ThisString : STRING, x : INTEGER) RETURNS STRING
```

returns rightmost `x` characters from `ThisString`.
Example: `RIGHT("ABCDEFGH", 3)` returns string `"FGH"`

```
TONUM(ThisString : STRING) RETURNS INTEGER
```

returns a numeric value equivalent to `ThisString`.
Example: `TONUM("1201")` returns integer value `1201`

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2017

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2021**

INSERT **2 hours**

0910143936-I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (PQ) 214716
© UCLES 2021

Note: An error occurs if a function call is not properly formed, or if the parameters are incorrect.

STRING Functions

`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"`

`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`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the lower case equivalent of `ThisChar`
Characters that are not upper case alphabetic are returned unchanged

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

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the upper case equivalent of `ThisChar`
Characters that are not lower case alphabetic are returned unchanged

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

`TO_UPPER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to upper case

Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`

`TO_LOWER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to lower case

Example: `TO_LOWER("JIM 803")` returns `"jim 803"`

`NUM_TO_STR(x : <data type>) RETURNS STRING`
returns a string representation of a numeric value
Note: `<data type>` may be `REAL` or `INTEGER`

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

```
STR_TO_NUM(x : <data type1>) RETURNS <data type2>
```
returns a numeric representation of a string
Note: `<data type1>` may be `CHAR` or `STRING`
Note: `<data type2>` may be `REAL` or `INTEGER`

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

```
IS_NUM(ThisString : STRING) RETURNS BOOLEAN
```
returns the value `TRUE` if `ThisString` represents a valid numeric value
Note: `<data type>` may be `CHAR` or `STRING`

Example: `IS_NUM("12.36")` returns `TRUE`
Example: `IS_NUM("-12.36")` returns `TRUE`
Example: `IS_NUM("12.3a")` returns `FALSE`

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

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

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

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

NUMERIC Functions

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

Example: `INT(27.5415)` returns `27`

```
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`

DATE Functions

Note: Date format is assumed to be DDMMYYYY unless otherwise stated.

```
DAY(ThisDate : DATE) RETURNS INTEGER
```
returns the current day number from `ThisDate`

Example: `DAY(4/10/2003)` returns `4`

```
MONTH(ThisDate : DATE) RETURNS INTEGER
```
returns the current month number from `ThisDate`

Example: `MONTH(4/10/2003)` returns `10`

`YEAR(ThisDate : DATE) RETURNS INTEGER`
returns the current year number from `ThisDate`

Example: `YEAR(4/10/2003)` returns `2003`

`DAYINDEX(ThisDate : DATE) RETURNS INTEGER`
returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2, Tuesday = 3 etc.

Example: `DAYINDEX(12/05/2020)` returns `3`

`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE`
returns a variable of type `DATE`

`NOW() RETURNS DATE`
returns the current date

OTHER Functions

`EOF(FileName : STRING) RETURNS BOOLEAN`
returns `TRUE` if there are no more lines to be read from file `FileName`

Note: This function will generate an ERROR if the file is not already open in `READ` mode

OPERATORS

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: This operator may also be used to concatenate a character with a string
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example `10 DIV 3` evaluates to `3`

Note: An error is generated if an operator is used with a value or values of an incorrect type.

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.

© UCLES 2021

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

1621126030

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2016**

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 (NF/FD) 125122
© UCLES 2016

CAMBRIDGE International Examinations

There is an **Appendix** on page 18. Some questions will refer you to this information.

1 The items in the table below are individual statements in a generic programming language.

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

(a) (i) Show what type of programming construct each statement represents.

Complete the table by putting a tick (✓) in the appropriate column for each item.

Item	Statement	Selection	Iteration	Assignment
1	`MyScore = 65`			
2	`FOR IndexVal = 0 TO 99`			
3	`MyArray[3] = MID(MyString,3,2)`			
4	`IF MyScore >= 70 THEN`			
5	`ENDWHILE`			
6	`ELSE Message = "Error"`			

[6]

(ii) State the purpose of each statement in the table in **part (a)(i)**.

Do **not** use mathematical symbols in your descriptions.

Item	Purpose of statement
1	....
2	....
3	....
4	....
5	....
6	....

[6]

(iii) Evaluate the following expressions when `MyString` has the value `"Adaptive Maintenance"`.

Expression	Result
`'D' & RIGHT(MyString, 4)`	
`LEFT(RIGHT(MyString, 7), 3)`	

[2]

2 A team is designing a software system to monitor temperature in a process. To do this, the system needs to sample the temperature repeatedly. If the temperature exceeds a given threshold value, an alarm will sound.

The system is to be software-based. It will include a subroutine, `SampleTemp`, which samples the temperature and sets the alarm state to either ON or OFF.

The initial design stage will produce a prototype of `SampleTemp` with a user interface.
The structured English for this is:

1. IF the temperature does not exceed threshold value, SET alarm state to OFF
2. INPUT threshold value (to two decimal places)
3. INPUT sensor value (a whole number in the range 0 to 100)
4. MULTIPLY sensor value by conversion factor 1.135 to give temperature
5. IF temperature exceeds threshold value SET alarm state to ON
6. IF temperature exceeds threshold value OUTPUT message “Temperature Alarm”
7. IF temperature does not exceed threshold value OUTPUT message “Temperature OK”

(a) The procedure needs four variables. Complete the identifier table below for these variables.

Identifier	Data type	Description
`AlarmState`		....
`SensorValue`		....
`ThresholdValue`		....
`Temperature`		....

[4]

(b) Write the **pseudocode** equivalent of the structured English. Use the identifiers from the table in **part (a)**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

Question 3 begins on page 7.

3 A string encryption function is needed. The encryption uses a simple character-substitution method.

In this method, a new character substitutes for each character in the original string. This will create the encrypted string.

The substitution uses the 7-bit ASCII value for each character. This value is used as an index for a 1D array, `Lookup`, which contains the substitute characters.
`Lookup` contains an entry for each of the ASCII characters. It may be assumed that the original string and the substitute characters are all printable.

For example:

- 'A' has ASCII value 65
- Array element with index 65 contains the character 'Y' (the substitute character)
- Therefore, 'Y' substitutes for 'A'
- There is a different substitute character for every ASCII value

The programmer writes a function, `EncryptString`, to return the encrypted string. This function will receive two parameters, the original, `PlainText` string and the 1D array.

(a) The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode.

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

```
FUNCTION EncryptString(............................) RETURNS STRING

   DECLARE ................... , ...................... : CHAR

   DECLARE OldCharValue : ....................................

   DECLARE n : INTEGER

   DECLARE OutString : STRING

   ............................ //initialise the return string

   //loop through PlainText to produce OutString

   FOR n ← 1 TO ................ //from first to last character

      OldChar ← ...................//get next character

      OldCharValue ← ..............//find the ASCII value

      NewChar ← ...................//look up substitute character

      ............................//concatenate to OutString

   ENDFOR

   ..............................................................
ENDFUNCTION
```

[10]

(b) Additional code needs to be written to allow the user to change some of the characters in the array `Lookup`.

The user will input:

- the array start position
- the number of elements to change
- each new substitute character

At the end, the program will finally output a confirmation message.

The first version of the algorithm is represented by the flowchart on the following page.

(i) Write **program code** to declare the array `Lookup`.

Programming language .....

.....

.....

.....[2]

(ii) Write **program code** to implement the flowchart design.

In addition to the `Lookup` array, assume that the following variables have been declared:

```
StartPos, NumToChange, n, NewChar
```

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

START
Input StartPos
Input NumToChange
n ← 0
Loop
Output prompt for new substitute character
Input NewChar
Assign NewChar to current array element
n = NumToChange – 1?
No
Increment n
Yes
Output message stating number of elements changed
STOP

4 **(a)** Structured programming involves the breaking down of a problem into modules.

Give **two** reasons why this is done.

1 .....

.....

2 .....

.....

[2]

(b) A team needs to write a program to implement an online shopping system. Customers will access the program via a website.

Customers can search for items before adding them to a virtual shopping basket. When they have finished shopping, they pay for the items. The program provides output for the dispatch of the items.

Some of the key features of the system are as follows:

- a customer can add many items to the shopping basket
- payment may be either by credit or debit card, or by adding to a customer account
- the shop may dispatch the items in one or more packages

The structure chart below shows the program modules only.

(i) Draw on the chart, the symbols that represent the key features listed in **part (b)** above.

[3]

(ii) A section of the chart in **part (b)(i)** is shown below. It is to show the parameters passed between the Checkout and Card payment modules.

Name the three data items corresponding to the arrows.

Arrow	Data item
A	
B	
C	

[3]

Question 5 begins on page 13.

5 Toni has a large collection of jazz CDs that are stored in different places. She wants to record where the CDs are stored. She decides to write a program to do this.

The program must store the data in a file, `MyMusic`.

(a) **(i)** Why is a file needed?

.....

.....[1]

(ii) `MyMusic` is a text file with the data for each CD as one line of text.

Data for a typical CD are:

Title:	Kind of Green
Artist:	Miles Coltrane
Location:	Rack1-5

The line will be formed by concatenating the three data items.

For the example above, the line stored will be:

```
Kind of GreenMiles ColtraneRack1-5
```

Describe a problem that might occur when organising the data in this way.

.....

.....

.....

.....

Describe a possible solution.

.....

.....

.....

.....
[4]

(b) Toni must input the data into the file for all of her CDs.

A procedure, `InputData`, is needed to do this.

Toni designs the procedure and chooses the following identifiers:

Identifier	Data type
`CDTitle`	`STRING`
`CDArtist`	`STRING`
`CDLocation`	`STRING`

The procedure repeatedly performs the following steps:

- input a CD title (A rogue value of “##” is to be used to end the input)
- input the artist
- input the location
- create the text line
- write the text line to the file

When the rogue value is encountered the file is closed.

Write **program code** for the procedure `InputData`.

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

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

6 A string-handling function has been developed. The pseudocode for this function is shown below.

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

```
FUNCTION SSM(String1, String2 : STRING) RETURNS INTEGER
   DECLARE n, f, x, y : INTEGER

   n ← 0
   f ← 0

   REPEAT
      n ← n + 1
      x ← n
      y ← 1
      WHILE MID(String1, x, 1) = MID(String2, y, 1)

         IF y = LENGTH(String2)
            THEN
               f ← n
            ELSE
               x ← x + 1
               y ← y + 1
         ENDIF

      ENDWHILE

   UNTIL (n = LENGTH(String1)) OR (f <> 0)

   RETURN f

ENDFUNCTION
```

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

```
SSM("RETRACE", "RAC")
```

n	f	x	y	MID(String1, x, 1)	MID(String2, y, 1)
0	0				

[6]

(b) **(i)** Describe the purpose of function `SSM`.

.....

.....

.....

.....[2]

(ii) One of the possible return values from function `SSM` has a special meaning.

State the value and its meaning.

Value .....

Meaning .....
[2]

(iii) There is a problem with the logic of the pseudocode. This could generate a run-time error.

Describe the problem.

.....

.....

.....

.....[2]

Appendix

Built-in functions

In each function below, if the function call is not properly formed, the function returns an error.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`

returns the string of length `y` starting at position `x` from `ThisString`

Example: **`MID("ABCDEFGH", 2, 3)`** will return string **`"BCD"`**

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`

returns the leftmost `x` characters from `ThisString`

Example: **`LEFT("ABCDEFGH", 3)`** will return string **`"ABC"`**

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`

returns the rightmost `x` characters from `ThisString`

Example: **`RIGHT("ABCDEFGH", 3)`** will return string **`"FGH"`**

`ASC(ThisChar : CHAR) RETURNS INTEGER`

returns the ASCII value of character `ThisChar`

Example: **`ASC('W')`** will return **`87`**

`LENGTH(ThisString : STRING) RETURNS INTEGER`

returns the integer value representing the length of string `ThisString`

Example: **`LENGTH("Happy Days")`** will return **`10`**

String operator

`&` operator

concatenates (joins) two strings

Example: **`"Summer" & " " & "Pudding"`** produces **`"Summer Pudding"`**

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2016

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2022**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (RW/CGW) 303754/3
© UCLES 2022

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** A programmer draws a program flowchart to show the sequence of steps required to solve a problem.

Give the technical term for a sequence of steps that describe how to solve a problem.

.....

..... [1]

(b) The table lists some of the variables used in a program.

(i) Complete the table by writing the most appropriate data type for each variable.

Variable	Use of variable	Data type
`Temp`	Stores the average temperature	
`PetName`	Stores the name of my pet	
`MyDOB`	To calculate the number of days until my next birthday	
`LightOn`	Stores state of light; light is only on or off	

[4]

(ii) One of the names used for a variable in the table in part **1(b)(i)** is not an example of good practice.

Identify the variable and give a reason why it is **not** good practice to use that name.

Variable .....

Reason .....

.....

.....

[2]

(c) Complete the table by evaluating each expression.

Expression	Evaluation
`INT((31 / 3) + 1)`	
`MID(TO_UPPER("Version"), 4, 2)`	
`TRUE AND (NOT FALSE)`	
`NUM_TO_STR(27 MOD 3)`	

[4]

2 Examine the following state-transition diagram.

(a) Complete the table with reference to the diagram.

	Answer
The number of different inputs	
The number of different outputs	
The single input value that could result in S4	

[3]

(b) The initial state is S1.

Complete the table to show the inputs, outputs and next states.

Input	**Output**	**Next state**
Button-Y		
	none	
Button-Z		S2
	none	

[4]

3 The manager of a cinema wants a program to allow users to book seats. The cinema has several screens. Each screen shows a different film.

(a) Decomposition will be used to break the problem down into sub-problems.

Describe **three** program modules that could be used in the design.

Module 1 .....

.....

.....

Module 2 .....

.....

.....

Module 3 .....

.....

.....
[3]

(b) Two types of program modules may be used in the design of the program.

Identify the type of program module that should be used to return a value.

..... [1]

BLANK PAGE

4 A stack is created using a high-level language. Memory locations 200 to 207 are to be used to store the stack.

The following diagram represents the current state of the stack.

`TopOfStack` points to the last value added to the stack.

Stack		Pointer
Memory location	**Value**	
200		
201		
202		
203	'F'	← `TopOfStack`
204	'C'	
205	'D'	
206	'E'	
207	'H'	

(a) Complete the following table by writing the answers.

	Answer
The value that has been on the stack for the longest time.	
The memory location pointed to by `TopOfStack` if **three** POP operations are performed.	

[2]

(b) The following diagram shows the current state of the stack:

Stack		Pointer
Memory location	**Value**	
200		
201		
202	'W'	← `TopOfStack`
203	'Y'	
204	'X'	
205	'Z'	
206	'N'	
207	'P'	

The following operations are performed:

POP
POP
PUSH 'A'
PUSH 'B'
POP
PUSH 'C'
PUSH 'D'

Complete the diagram to show the state of the stack **after** the operations have been performed.

Stack		Pointer
Memory location	**Value**	
200		
201		
202		
203		
204		
205		
206		
207		

[4]

5 Each line of a text file contains data organised into fixed-length fields as shown:

<Field 1><Field 2><Field 3>

An algorithm is required to search for the first instance of a given value of Field 2 and, if found, to output the corresponding values for Field 1 and Field 3.

Describe the algorithm needed.

Do **not** include pseudocode statements in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

6 **(a)** An algorithm will:

- output **each** integer value between 100 and 200 that ends with the digit 7, for example, 107
- output a final count of the number of values that are output.

Write pseudocode for this algorithm.

Any variables used must be declared.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) Study the following pseudocode.

```
CASE OF MySwitch
   1: ThisChar ← 'a'
   2: ThisChar ← 'y'
   3: ThisChar ← '7'
   OTHERWISE: ThisChar ← '*'
ENDCASE
```

Write pseudocode with the same functionality **without** using a `CASE` structure.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

7 A string is a palindrome if it reads the same forwards as backwards.

The following strings are examples of palindromes:

"Racecar"
"madam"
"12344321"

Upper-case and lower-case characters need to be treated the same. For example, 'A' is equivalent to 'a'.

(a) A function `IsPalindrome()` will take a string parameter. The function will return `TRUE` if the string is a palindrome and will return `FALSE` if the string is not a palindrome.

Write pseudocode for `IsPalindrome()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) Strings may consist of several words separated by spaces.

For example, the string "never odd or even" becomes a palindrome if the spaces are removed.

The program flowchart represents an algorithm to produce a string `OutString` by removing all spaces from a string `InString`.

Complete the table by writing the text that should replace each of the labels **B**, **C**, **D**, **F** and **G**.

Note: the text may be written as a pseudocode statement.

Label	Text
A	Set `OutString` to `""`
B	
C	
D	
E	Set `Index` to `Index + 1`
F	
G	

[4]

8 A program allows a user to save passwords used to login to websites. A stored password is inserted automatically when the user logs into the corresponding website.

A student is developing a program to generate a password. The password will be of a fixed format, consisting of **three groups of four** alphanumeric characters. The groups are separated by the hyphen character '-'.

An example of a password is: "FxAf-3haV-Tq49"

A global 2D array `Secret` of type `STRING` stores the passwords together with the website domain name where they are used. `Secret` contains 1000 elements organised as 500 rows by 2 columns.

Unused elements contain the empty string (`""`). These may occur anywhere in the array.

An example of a part of the array is:

Array element	Value
`Secret[27, 1]`	"www.thiswebsite.com"
`Secret[27, 2]`	"••••••••••••••"
`Secret[28, 1]`	"www.thatwebsite.com"
`Secret[28, 2]`	"••••••••••••••"

Note:
- For security, passwords are stored in an encrypted form, shown as "••••••••••••••" in the example.
- The passwords cannot be used without being decrypted.
- Assume that the encrypted form of a password will **not** be an empty string.

The programmer has started to define program modules as follows:

Module	Description
`RandomChar()`	• Generates a single random character from within one of the following ranges: ○ 'a' to 'z' ○ 'A' to 'Z' ○ '0' to '9' • Returns the character
`Encrypt()`	• Takes a password as a parameter of type string • Returns the encrypted form of the password as a string
`Decrypt()`	• Takes an encrypted password as a parameter of type string • Returns the decrypted form of the password as a string

For reference, relevant ASCII values are as follows:

Character range	ASCII range
'a' to 'z'	97 to 122
'A' to 'Z'	65 to 90
'0' to '9'	48 to 57

(a) Write pseudocode for module `RandomChar()`.

You may wish to refer to the **insert** for a description of the `CHR()` function. Other functions may also be required.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) A new module is defined as follows:

Module	Description
`FindPassword()`	• Takes a website domain name as a parameter of type string • Searches for the website domain name in the array `Secret` • If the website domain name is found, the decrypted password is returned • If the website domain name is not found, a warning message is output, and an empty string is returned

Write pseudocode for module `FindPassword()`.

Assume that modules `Encrypt()` and `Decrypt()` have already been written.

..... [7]

(c) The modules `Encrypt()` and `Decrypt()` are called from several places in the main program.

Identify a method that could have been used to test the main program before these modules were completed. Describe how this would work.

Method .....

Description .....

.....

.....

..... [3]

(d) A validation function is written to check that the passwords generated are valid.

To be valid, each password must:

- be 14 characters long
- be organised as three groups of four case-sensitive alphanumeric characters. The groups are separated by hyphen characters
- not include any duplicated characters, except for the hyphen characters.

Note: lower-case and upper-case characters are not the same. For example, 'a' is not the same as 'A'.

Give **two** password strings that could be used to test different areas of the validation rules.

Password 1 .....

Password 2 .....
[2]

(e) The `RandomChar()` module is to be modified so that alphabetic characters are generated twice as often as numeric characters.

Describe how this might be achieved.

.....

.....

.....

.....

.....

..... [3]

BLANK PAGE

BLANK PAGE

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

6683806624

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
© UCLES 2019

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]

(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]

© UCLES 2019

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`

© UCLES 2019

BLANK PAGE

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.

© UCLES 2019

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2023**

INSERT **2 hours**

1281968342-I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (EF) 312085/3
© UCLES 2023

An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

- A string of length 1 may be either of type `CHAR` or `STRING`
- A `CHAR` may be assigned to, or concatenated with, a `STRING`
- A `STRING` of length greater than 1 cannot be assigned to a `CHAR`

`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"`
`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 string `"BCD"`
`LENGTH(ThisString : STRING) RETURNS INTEGER` returns the integer value representing the length of `ThisString` Example: `LENGTH("Happy Days")` returns `10`

`TO_UPPER(x : <datatype>) RETURNS <datatype>`
`<datatype>` may be `CHAR` or `STRING`
returns an object of type `<datatype>` formed by converting all characters of `x` to upper case.
Examples:
• `TO_UPPER("Error 803")` returns `"ERROR 803"`
• `TO_UPPER('a')` returns `'A'`

`TO_LOWER(x : <datatype>) RETURNS <datatype>`
`<datatype>` may be `CHAR` or `STRING`
returns an object of type `<datatype>` formed by converting all characters of `x` to lower case.
Examples:
• `TO_LOWER("JIM 803")` returns `"jim 803"`
• `TO_LOWER('W')` returns `'w'`

`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>`
returns a string representation of a numeric value.
`<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING`
Example: `NUM_TO_STR(87.5)` returns `"87.5"`

`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>`
returns a numeric representation of a string.
`<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER`
Example: `STR_TO_NUM("23.45")` returns `23.45`

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN`
returns `TRUE` if `ThisString` represents a valid numeric value.
`<datatype>` may be `CHAR` or `STRING`
Example: `IS_NUM("-12.36")` returns `TRUE`

`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`RAND(x : INTEGER) RETURNS REAL` returns a real number in the range `0` to `x` (**not** inclusive of `x`). Example: `RAND(87)` may return `35.43`

Date Functions

Date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the day index number from `ThisDate` where Sunday = 1, Monday = 2 etc. Example: `DAYINDEX(09/05/2023)` returns 3
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a value of type `DATE` with the value of `<Day>/<Month>/<Year>` Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26/10/2003
`TODAY() RETURNS DATE` returns a value of type `DATE` corresponding to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` The function will generate an error if the file is not already open in `READ` mode.

Operators

An error will be generated if an operator is used with a value or values of an incorrect type.

`&`	concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` The operator may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values. Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values. Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value. Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another. Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another. Example `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	used to compare two items of the same type.
	evaluates to `TRUE` if the condition is true, otherwise evaluates to `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when variable `Count` contains the value 4

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2021**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (LK/CGW) 213669/3
© UCLES 2021

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** A programmer applies decomposition to a problem that she has been asked to solve.

Describe decomposition.

.....

.....

.....

.....

.....

..... [2]

(b) The following pseudocode assigns a value to an element of an array:

```
ThisArray[n] ← 42
```

Complete the following table by writing the answer for each row.

	Answer
The number of dimensions of `ThisArray`	
The technical terms for minimum and maximum values that the variable `n` may take	
The technical term for the variable `n` in the pseudocode statement	

[3]

(c) Complete the pseudocode expressions so that they evaluate to the values shown.

Any functions and operators used must be defined in the **insert**.

Expression	**Evaluates to**
..... ('C')	67
2 * ..... ("27")	54
..... (27 / .....)	13
"Sub" & ..... ("Abstraction" , ... , ...)	"Subtract"

[4]

(d) Evaluate the expressions given in the following table. The variables have been assigned values as follows:

```
PumpOn ← TRUE
PressureOK ← TRUE
HiFlow ← FALSE
```

Expression	Evaluates to
`PressureOK AND HiFlow`	
`PumpOn OR PressureOK`	
`NOT PumpOn OR (PressureOK AND NOT HiFlow)`	
`NOT (PumpOn OR PressureOK) AND NOT HiFlow`	

[2]

2 **(a)** An algorithm will:

1. input an integer value
2. jump to step 6 if the value is zero
3. sum and count the positive values
4. sum and count the negative values
5. repeat from step 1
6. output the two sum values and the two count values.

Draw a program flowchart on the following page to represent the algorithm.

Note that variable declarations are not required in program flowcharts.

[5]

(b) A software company is working on a project to develop a website for a school.

The school principal has some ideas about the appearance of the website but is unclear about all the details of the solution. The principal would like to see an initial version of the website.

(i) Identify a life cycle method that would be appropriate in this case.

Give a reason for your choice.

Life cycle method .....

Reason .....

.....

.....

.....
[2]

(ii) The website project has progressed to the design stage.

State **three** activities that will take place during the design stage of the program development life cycle.

1 .....

2 .....

3 .....
[3]

BLANK PAGE

3 A programmer is writing a program to help manage clubs in a school.

Data will be stored about each student in the school and each student may join up to three clubs.

The data will be held in a record structure of type `Student`.

The programmer has started to define the fields that will be needed as shown in the following table.

Field	Typical value	Comment
`StudentID`	`"CF1234"`	Unique to each student
`Email`	`"Carmen47@xyzmail.com"`	Contains letters, numbers and certain symbols
`Club_1`	`1`	Any value in the range 1 to 99 inclusive
`Club_2`	`14`	Any value in the range 1 to 99 inclusive
`Club_3`	`27`	Any value in the range 1 to 99 inclusive

(a) **(i)** Write pseudocode to declare the record structure for type `Student`.

.....

.....

.....

.....

.....

.....

..... [3]

(ii) A 1D array `Membership` containing 3000 elements will be used to store the student data.

Write pseudocode to declare the `Membership` array.

.....

..... [2]

(iii) Some of the elements of the array will be unused.

Give an **appropriate** way of indicating an unused array element.

.....

..... [1]

(iv) Some students are members of less than three clubs.

State **one** way of indicating an unused club field.

.....

..... [1]

(b) A procedure `GetIDs()` will:

- prompt and input the number of a club
- output the `StudentID` of all the students who are members of that club
- output a count of all students in the given club.

Write pseudocode for the procedure `GetIDs()`.

..... [7]

4 The following is a procedure design in pseudocode.

Line numbers are given for reference only.

```
10  PROCEDURE Check(InString : STRING)
11     DECLARE Odds, Evens, Index : INTEGER
12
13     Odds ← 0
14     Evens ← 0
15     Index ← 1
16
17     WHILE Index <= LENGTH(InString)
18        IF STR_TO_NUM(MID(InString, Index, 1)) MOD 2 <> 0 THEN
19           Odds ← Odds + 1
20        ELSE
21           Evens ← Evens + 1
22        ENDIF
23        Index ← Index + 1
24     ENDWHILE
25
26     CALL Result(Odds, Evens)
27  ENDPROCEDURE
```

(a) Complete the following table by giving the answers, using the given pseudocode.

	Answer
A line number containing a variable being incremented	
The type of loop structure	
The number of functions used	
The number of parameters passed to `STR_TO_NUM()`	
The name of a procedure other than `Check()`	

[5]

(b) The pseudocode includes several features that make it easier to read and understand.

Identify **three** of these features.

1 .....

2 .....

3 .....

[3]

(c) **(i)** The loop structure used in the pseudocode is not the most appropriate.

State a more appropriate loop structure **and** justify your choice.

Loop structure .....

Justification .....

.....

.....
[2]

(ii) The appropriate loop structure is now used. Two lines of pseudocode are changed and two lines are removed.

Write the line numbers of the two lines that are removed.

.....

..... [1]

5 A company has several departments. Each department stores the name, email address and the status of each employee in that department in its own text file. All text files have the same format.

Employee details are stored as three separate data strings on three consecutive lines of the file. An example of the first six lines of one of the files is as follows:

File line	Comment
1	First employee name
2	First email address
3	First employee status
4	Second employee name
5	Second email address
6	Second employee status

A procedure `MakeNewFile()` will:

- take three parameters as strings:
  - an existing file name
  - a new file name
  - a search status value
- create a new text file using the new file name
- write all employee details to the new file where the employee status is **not** equal to the search status value
- count the number of sets of employee details that were in the original file
- count the number of sets of employee details that were written to the new file
- produce a summary output.

An example summary output is as follows:

```
File Marketing contained 54 employee details
52 employee sets of details were written to file NewMarketingList
```

(a) Write pseudocode for the procedure `MakeNewFile()`.

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) An alternative format could be used for storing the data.

A text file will still be used.

(i) Describe the alternative format.

.....

..... [1]

(ii) State **one** advantage **and one** disadvantage of the alternative format.

Advantage .....

.....

Disadvantage .....

.....
[2]

BLANK PAGE

6 A mobile phone has a touchscreen. The screen is represented by a grid, divided into 800 rows and 1280 columns.

The grid is represented by a 2D array `Screen` of type `INTEGER`. An array element will be set to 0 unless the user touches that part of the screen.

Many array elements will set to 1 by a single touch of a finger or a stylus.

The following diagram shows a simplified touchscreen. The dark line represents a touch on the screen. All grid elements that are wholly or partly inside the outline will be set to 1. These elements are shaded.
The element shaded in black represents the centre point of the touch.

A program is needed to find the coordinates (the row and column) of the centre point. The centre point on the diagram shown is row 6, column 11.

Assume:

- the user may only touch one area at a time
- screen rotation does not affect the touchscreen.

The programmer has decided to use global values `CentreRow` and `CentreCol` as coordinate values for the centre point.

The programmer has started to define program modules as follows:

Module	Description
`FirstRowSet()`	• Searches for the first row that has an array element set to 1 • Returns the index of that row (1 is the first row) • Returns −1 if there are no elements set to 1
`LastRowSet()`	• Searches for the last row that has an array element set to 1 • Returns the index of that row • Returns −1 if there are no elements set to 1
`FirstColSet()`	• Searches for the first column that has an array element set to 1 • Returns the index of that column (1 is the first column) • Returns −1 if there are no elements set to 1
`LastColSet()`	• Searches for the last column that has an array element set to 1 • Returns the index of that column • Returns −1 if there are no elements set to 1

© UCLES 2021

(a) Write efficient pseudocode for the module `FirstRowSet()`.

..... [7]

(b) Describe a feature of your solution to **part (a)** that indicates the pseudocode represents an efficient algorithm.

.....

.....

.....

..... [2]

(c) The programmer decides to produce a **single** search module `FindSet()`, which will be able to perform each of the individual searches performed by the first four modules in the table.

(i) Outline the changes needed to convert one of the existing modules into this single module.

.....

.....

.....

.....

.....

..... [2]

(ii) Give one possible advantage **and** one possible disadvantage of combining the four searches into a single module.

Advantage .....

.....

Disadvantage .....

.....

[2]

(d) An additional module `GetCentre()` is defined as follows:

Module	Description
`GetCentre()`	• Calculates the coordinates of the centre point • Uses the four modules: `FirstRowSet()`, `LastRowSet()`, `FirstColSet()`, `LastColSet()` • Assigns values to `CentreRow` and `CentreCol` • Sets `CentreRow` to −1 if no screen touch is detected

Write pseudocode for the module `GetCentre()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

5904136477

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2022**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (LK/JG) 302746/2
© UCLES 2022

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** A programmer is developing an algorithm to solve a problem. Part of the algorithm would be appropriate to implement as a subroutine (a procedure or a function).

(i) State **two** reasons why the programmer may decide to use a subroutine.

1 .....

.....

2 .....

.....
[2]

(ii) A procedure header is shown in pseudocode:

```
PROCEDURE MyProc(Count : INTEGER, Message : STRING)
```

Give the correct term for the identifiers `Count` and `Message` **and** explain their use.

Term .....

Use .....

.....

.....

.....
[2]

(b) The algorithm in **part (a)** is part of a program that will be sold to the public.
All the software errors that were identified during in-house testing have been corrected.

Identify **and** describe the additional test stage that may be carried out before the program is sold to the public.

Test stage .....

Description .....

.....

.....

.....

.....

.....
[4]

(c) Part of an identifier table is shown:

Variable	Type	Example value
`FlagDay`	`DATE`	`23/04/2004`
`CharList`	`STRING`	`"ABCDEF"`
`Count`	`INTEGER`	`29`

Complete the table by evaluating each expression using the example values.

Expression	Evaluation
`MID(CharList, MONTH(FlagDay), 1)`	
`INT(Count / LENGTH(CharList))`	
`(Count >= 29) AND (DAY(FlagDay) > 23)`	

[3]

2 **(a)** An algorithm will process data from a test taken by a group of students. The algorithm will prompt and input the name and test mark for each of the 35 students.

The algorithm will add the names of all the students with a test mark of less than 20 to an existing text file `Support_List.txt`, which already contains data from other group tests.

(i) Describe the steps that the algorithm should perform.

Do **not** include pseudocode statements in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(ii) Explain why it may be better to store the names of the students in a file rather than in an array.

.....

.....

..... [1]

(iii) Explain why `WRITE` mode cannot be used in the answer to **part 2(a)(i)**.

.....

.....

..... [1]

(b) Examine the following state-transition diagram.

Complete the table to show the inputs, outputs and next states.

Input	Output	Next state
		S1
Input-A		
		S2
	Output-W	
	Output-W	

[4]

3 A stack is used in a program to store string data which needs to be accessed in several modules.

(a) A stack is an example of an Abstract Data Type (ADT).

Identify **one other** example of an ADT **and** describe its main features.

Example .....

Features .....

.....

.....

.....

.....

.....
[3]

(b) Explain how the stack can be implemented using an array.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(c) A second stack is used in the program. The diagram below shows the initial state of this stack. Value X is at the top of the stack and was the last item added.

Upper-case letters are used to represent different data values.

Stack operations are performed in three groups as follows:

Group 1:

```
PUSH D
PUSH E
```

Group 2:

```
POP
POP
POP
```

Group 3:

```
PUSH A
PUSH B
POP
PUSH C
```

Complete the diagram to show the state of the stack **after** each group of operations has been performed.

Include the current stack pointer (SP) **after** each group.

Memory location	**Initial state**		**After Group 1**	**After Group 2**	**After Group 3**
957					
956					
955					
954					
953	X	←SP			
952	Y				
951	Z				
950	P				

[5]

BLANK PAGE

4 The program flowchart represents a simple algorithm.

(a) Write the equivalent pseudocode for the algorithm represented by the flowchart.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) Give the name of the iterative construct in the flowchart.

..... [1]

5 Examine the following pseudocode.

```
IF A = TRUE THEN
   IF B = TRUE THEN
      IF C = TRUE THEN
         CALL Sub1()
      ELSE
         CALL Sub2()
      ENDIF
   ENDIF
ELSE
   IF B = TRUE THEN
      IF C = TRUE THEN
        CALL Sub4()
      ELSE
        CALL Sub3()
      ENDIF
   ELSE
      IF C = FALSE THEN
        CALL Sub3()
      ELSE
        CALL Sub4()
      ENDIF
   ENDIF
ENDIF
```

A programmer wants to re-write the pseudocode as **four** separate `IF...THEN...ENDIF` statements, each containing a single `CALL` statement. This involves writing a single, simplified logic expression as the condition in each statement.

Write the amended pseudocode.

1 .....

.....

.....

2 .....

.....

.....

3 .....

.....

.....

4 .....

.....

.....

[4]

BLANK PAGE

6 **(a)** The factorial of an integer number is the product of all the integers from that number down to 1.

In general, the factorial of n is n × (n−1) × ... × 2 × 1

For example, the factorial of 5 is 5 × 4 × 3 × 2 × 1 = 120

In this question, n will be referred to as the `BaseNumber`.

A function `FindBaseNumber()` will:

- be called with a positive, non-zero integer value as a parameter
- return `BaseNumber` if the parameter value is the factorial of the `BaseNumber`
- return −1 if the parameter value **is not** a factorial.

For example:

Parameter value	Value returned
120	5
12	−1
6	3
1	1

`FindBaseNumber(12)` will return −1 because 12 is not a factorial.

You may use the rest of this page for rough working.

Write pseudocode for the function `FindBaseNumber()`.

..... [7]

(b) A program is written to allow a user to input a sequence of values to be checked using the function `FindBaseNumber()`.

The user will input one value at a time. The variable used to store the user input has to be of type string because the user will input 'End' to end the program.

Valid input will be converted to an integer and passed to `FindBaseNumber()` and the return value will be output.

Complete the table by giving **four** invalid strings that may be used to test distinct aspects of the required validation. Give the reason for your choice in each case.

Input	Reason for choice
	....
	....
	....
	....

[4]

BLANK PAGE

7 A teacher is designing a program to perform simple syntax checks on programs written by students.

Two global 1D arrays are used to store the syntax error data. Both arrays contain 500 elements.

- Array `ErrCode` contains integer values that represent an error number in the range 1 to 800.
- Array `ErrText` contains string values that represent an error description.

The following diagram shows an example of the arrays.

Index	ErrCode	ErrText
1	10	"Invalid identifier name"
2	20	"Bracket mismatch"
3	50	"Undeclared variable"
4	60	"Type mismatch in assignment"
…		
500	999	<Undefined>

Note:

- There may be less than 500 error numbers so corresponding elements in both arrays may be unused. Unused elements in `ErrCode` have the value `999`. The value of unused elements in `ErrText` is undefined.
- Values in the `ErrCode` array are stored in ascending order but not all values may be present, for example, there may be no error code `31`.

The teacher has defined two program modules as follows:

Module	Description

`OutputError()`	• takes two parameters as integers:
	○ a line number in the student's program
	○ an error number
	• searches for the error number in the `ErrCode` array:
	○ if found, outputs the corresponding error description and the line number, for example:
	`"Bracket mismatch on line 34"`
	○ if not found, outputs the line number and a warning, for example:
	`"Unknown error on line 34"`

`SortArrays()`	sorts the arrays into ascending order of `ErrCode`

(a) Write **efficient** pseudocode for module `OutputError()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) Write an **efficient** bubble sort algorithm in pseudocode for module `SortArrays()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(c) Two 1D arrays were described at the beginning of the question. Both arrays contain 500 elements.

- Array `ErrCode` contains integer values that represent an error number in the range 1 to 800.
- Array `ErrText` contains string values that represent an error description.

The two arrays will be replaced by a single array. A user-defined data type (record structure) has been declared as follows:

```
TYPE ErrorRec
   DECLARE ErrCode : STRING
   DECLARE ErrText : STRING
ENDTYPE
```

(i) State the error in the record declaration.

.....

..... [1]

(ii) State **two** benefits of using the single array of the user-defined data type.

1 .....

.....

2 .....

.....
[2]

(iii) Write the declaration for the single array in pseudocode.

..... [1]

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

2720504716

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2018**

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 **16** printed pages.

DC (NH/CT) 150983/3
© UCLES 2018

CAMBRIDGE International Examinations

Question 1 begins on the next 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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2018

1 **(a)** The following table contains statements written in pseudocode.

Show the type of programming construct each statement represents.

Put a tick (✓) in the appropriate column for each statement.

Statement	Selection	Repetition (Iteration)	Assignment
`Index ← Index + 5`			
`FOR Count ← 1 TO 100`			
`TempValue[Index] ← ReadValue(SensorID)`			
`IF Index < 30`			
`UNTIL DayNumber > 7`			
`OTHERWISE OUTPUT "ERROR"`			

[6]

(b) **(i)** The following table contains statements written in pseudocode.

Give the most appropriate data type for the variable used in each statement.

Statement	Data type
`Revision ← 'B'`	
`MaxValue ← 13.3`	
`ArrayFull ← TRUE`	
`Activity ← "Design"`	
`NumberOfEdits ← 270`	

[5]

(ii) The following table contains statements written in pseudocode.

Complete the table by evaluating each expression using the values from **part (b)(i)**.
If any expression is invalid, write “ERROR” in the **Evaluates to** column.

For the built-in functions list, refer to the **Appendix** on page 16.

Expression	Evaluates to
`MID(Activity, 3, 4) & "ature"`	
`INT(MaxValue * 2)`	
`ArrayFull AND NumberOfEdits < 300`	
`ASC(Revision + 1)`	
`Activity = "Testing" OR Revision = 'A'`	

[5]

2 Shop customers have a discount card with a unique card number. Customers collect points each time they buy items. The number of points they collect depends on:

- the total amount they spend
- the number of points already collected.

The function `CalcPoints()` takes the card number and the total amount spent as parameters. It returns the number of new points collected. A flowchart for the function is shown.

The function uses the following variables and functions.

Identifier	Data type	Description
`CardNum`	`STRING`	A numeric string representing the unique card number
`OldPoints`	`INTEGER`	The number of points already collected
`NewPoints`	`INTEGER`	The number of new points collected
`Total`	`REAL`	The amount spent
`GetPoints()`	`FUNCTION`	Takes the card number as a parameter and returns the number of points already collected
`INT()`	`FUNCTION`	Refer to the **Appendix** on page 16

(a) **(i)** Write **pseudocode** for the `CalcPoints()` function.

Your solution should follow the flowchart as closely as possible.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[7]

(ii) The value of the total amount spent is calculated by an Electronic Point Of Sale (EPOS) system. This system does not have the prices of all items. For these items, a valid total amount has to be entered manually.

A function, `GetTotal()`, prompts the user to input this value.

If the user enters a valid value greater than 0 and less than 10 000, the function returns the value. The function prompts the user to re-enter the value each time the user enters an invalid value.

Write **pseudocode** to complete the `GetTotal()` function.

```
FUNCTION GetTotal() RETURNS REAL
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
ENDFUNCTION
```

[5]

(b) The function `CalcPoints()` is written in a high-level language. It has been checked and it does not contain any syntax or logic errors.

(i) Name **and** describe **one** other type of error that the high-level language code could contain.

Name .....

Description .....

.....

.....
[2]

(ii) The function `CalcPoints()` is tested using white-box testing.

State **two** different values of `Total` that could be used to test different paths through the algorithm. Justify your choices.

Value .....

Justification .....

.....

.....

Value .....

Justification .....

.....

.....
[4]

3 **(a)** Programming is sometimes referred to as a **transferable skill**.

You are asked to work on a program written in a language you are not familiar with.

Explain how **transferable skills** would help you work on the program.

.....

.....

.....

.....[2]

(b) Stepwise refinement is often used in the development of an algorithm.

Describe **stepwise refinement**.

.....

.....

.....

.....[2]

(c) A program needs to search through 1000 elements of an unsorted array to find a given value.

The program will output:

- either the position in the array of the value
- or the message "Not Found"

Outline the steps the program needs to follow.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

Question 4 begins on the next page.

4 Part of a program written in pseudocode is shown.

```
010 DECLARE NextArrayElement : INTEGER
…

100 FUNCTION ScanFile(SearchString: STRING) RETURNS INTEGER
101
102    DECLARE FileData : STRING
103    DECLARE FileLine : INTEGER
104
105    NextArrayElement ← 1
106    FileLine ← 1
107    FileData ← ReadFileLine("DataFile.txt", FileLine)
108
109    WHILE FileData <> ""
110        IF LEFT(FileData, 7) = SearchString
111          THEN
112            ResultArray[NextArrayElement] ← FileData
113            NextArrayElement ← NextArrayElement + 1
114        ENDIF
115        FileLine ← FileLine + 1
116        FileData ← ReadFileLine("DataFile.txt", FileLine)
117    ENDWHILE
118
119    CALL ScanCompleted()
120    RETURN FileLine
121
122 ENDFUNCTION
```

(a) **(i)** Examine the pseudocode. Complete the following table.

	Answer
The identifier name of a local variable	
The identifier name of a user-defined procedure	
The identifier name of a user-defined function	
The number of dimensions of `ResultArray`	
The scope of `FileData`	

[5]

(ii) Describe in detail the purpose of lines `109` to `117` in the `ScanFile()` function. Do **not** use pseudocode in your answer.

.....

.....

.....

.....

.....

.....[4]

(b) The function `ScanFile()` is to be amended so that the first 7 characters of `FileData` are **not** written to the array.

Write **program code** to implement this **amended** `ScanFile()` function.

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]

(c) **(i)** State how structured programming languages support the implementation of sub-tasks.

.....[1]

(ii) State a benefit of using sub-tasks.

.....

.....[1]

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

Write **program code** to declare `ResultArray` and set all elements to the value `"NO DATA"`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....[3]

Question 5 begins on the next page.

5 The procedure `LineNumber()` will:

- read data from a text file
- modify each line by adding a line number, and the string (`": "`)
- output each modified line.

For example, when the procedure reads `MyFile.txt`, the output is:

```
10: <First line of MyFile.txt>
15: <Second line of MyFile.txt>
20: <Third line of MyFile.txt>
…
350: <Last line of MyFile.txt>
```

The procedure takes three parameters:

Identifier	Data type	Description
FileName	`STRING`	The name of the text file
StartNumber	`INTEGER`	The first line number to be added
StepNumber	`INTEGER`	The line number increment

In this example, the procedure call would be:

```
CALL LineNumber("MyFile.txt", 10, 5)
```

After every 20 lines, the program outputs a message asking whether the user wants to continue.

The program ends when the user enters an `'N'` or the end of file is reached.

Write **pseudocode** for the `LineNumber()` procedure.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

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 string `"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 string `"ABC"`

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

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

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

Example: If `x` has the value `87.5` then `NUM_TO_STRING(x)` will return `"87.5"`
Note: This function will also work if `x` is of type integer.

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

Example: `INT(27.5415)` returns `27`

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

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

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`

© UCLES 2018

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2020**

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 **20** pages. Blank pages are indicated.

DC (LK) 188573/3
© UCLES 2020

1 **(a)** Translation is one stage of the program development cycle.

State **three** other stages.

1 .....

2 .....

3 .....
[3]

(b) Define the following types of maintenance.

Corrective maintenance .....

.....

.....

Adaptive maintenance .....

.....

.....
[2]

(c) Experienced programmers have a **transferable skill**.

Explain how this skill might be useful for a programmer.

.....

.....

.....

..... [2]

(d) Jackie has written a program and has used the identifier names `I1`, `I2`, and `I3`.

Explain why this is not good practice.

.....

.....

.....

..... [2]

(e) A pseudocode algorithm assigns values to three variables as follows:

```
GateOpen ← FALSE
Alarm ← TRUE
PowerFail ← TRUE
```

Evaluate the expressions given in the following table:

Expression	Evaluates to
`Alarm OR NOT PowerFail`	
`NOT (Alarm AND PowerFail)`	
`(GateOpen OR Alarm) AND PowerFail`	
`(GateOpen AND Alarm) OR NOT PowerFail`	

[2]

2 **(a)** User names are stored in a text file. Each line of the file represents one name. Before a new user name can be issued, a check has to be made to ensure that the new name is unique.

Use **structured English** to describe an algorithm that would prompt and input a new user name and output a message to indicate whether or not it is unique.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) Complete the pseudocode expressions in the following table.

Use **only** the functions and operators described in the **Appendix** on pages 18–19.

Expression	Evaluates to
..... ("Stepwise" , ...) & "art"	"Start"
..... ("Concatenate" , ... , ...)	"ten"
2 * ..... ("Kipper")	12
TRUE ..... FALSE	TRUE
..... (9, 2)	1

[5]

© UCLES 2020

(c) Study the following pseudocode.

Line numbers are given for reference only.

```
01  PROCEDURE StringClean(InString : STRING)
02
03     DECLARE NextChar : CHAR
04     DECLARE OutString : STRING
05     DECLARE Index : INTEGER
06
07     OutString ← ""
08
09     FOR Index ← 1 TO LENGTH(InString)
10
11        NextChar ← MID(InString, Index, 1)
12        NextChar ← LCASE(NextChar)
13
14        IF NextChar >= 'a' AND NextChar <= 'z'
15           THEN
16              OutString ← OutString & NextChar
17        ENDIF
18
19     ENDFOR
20
21     OUTPUT OutString
22
23  ENDPROCEDURE
```

Complete the following table by entering an appropriate answer.

	Answer
The name for the type of loop used	
A line number of a selection statement	
The scope of `OutString`	
The name of a function that is called	
A line number containing a logical operator	

[5]

3 The procedure `OutputLines()` outputs a number of lines from a text file.

An example of the use of the procedure is given by the following pseudocode:

```
CALL OutputLines(FileName, StartLine, NumberLines)
```

Parameter	Data type	Description
`FileName`	`STRING`	The name of the text file
`StartLine`	`INTEGER`	The number of the first line to be output
`NumberLines`	`INTEGER`	The number of lines to be output

The procedure is tested using the file `MyFile.txt` that contains 100 lines of text.

The procedure gives the expected result when called as follows:

```
CALL OutputLines("MyFile.txt", 1, 10)
```

(a) The procedure is correctly called with three parameters of the appropriate data types, but the procedure does not give the expected result.

Give **three different** reasons why this might happen.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(b) Write **program code** for the procedure `OutputLines()`.

Note: Parameter validation is **not** necessary.

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) A program is compiled without producing any errors.

(i) Describe **one** type of error that the program could still contain.

.....

.....

.....

..... [2]

(ii) Give **two** techniques that may be used to identify an error of the type given in **part (c)(i)**.

Technique 1 .....

.....

Technique 2 .....

.....
[2]

(d) State **two** reasons why the use of library subroutines can be a benefit in program development.

1 .....

.....

2 .....

.....
[2]

BLANK PAGE

4 A function, `FormOut()`, takes an integer parameter in the range 0 to 999 999 and returns a formatted string depending on two other parameter values.

Formatting may incorporate the use of:

- A prefix string to be added before the integer value (e.g. `'$'` or `"Total: "`)
- A comma as a thousand-separator (e.g. `"1,000"`)

The function will be called as follows:

```
MyString ← FormOut(Number, Prefix, AddComma)
```

Parameter	Data type	Description
`Number`	`INTEGER`	The positive integer value to be formatted.
`Prefix`	`STRING`	A string that will appear in front of the numeric value. Set to an empty string if no prefix is required.
`AddComma`	`BOOLEAN`	`TRUE` if a comma is required in the formatted string. `FALSE` if a comma is not required in the formatted string.

(a) Fill in the tables to show **two** tests that could be carried out to test **different** aspects of the function.

Give the expected result for each test.

TEST 1

Parameter	Value
`Number`	
`Prefix`	
`AddComma`	

Expected return string:

.....

TEST 2

Parameter	Value
`Number`	
`Prefix`	
`AddComma`	

Expected return string:

.....

[4]

(b) Write **pseudocode** for the function `FormOut()`.

Refer to the **Appendix** on pages 18–19 for the list of built-in functions and operators.

..... [8]

5 A message may contain several hashtags.

A hashtag is a string consisting of a hash character '#', followed by one or more alphanumeric characters.

A hashtag may be terminated by a space character, the start of the next hashtag, any other non-alphanumeric character, or by the end of the message.

For example, the following message contains three hashtags:

```
"#Error27 is the result of #PoorPlanning by the #Designer"
```

The hashtags in the message are `"#Error27"`, `"#PoorPlanning"` and `"#Designer"`.

A program is being developed to process a message and extract each hashtag.

A global 1D array of strings, `TagString`, will store each hashtag in a single element. Unused array elements will contain an empty string. The array will contain 10 000 elements.

A developer has started to define the modules as follows:

Module	Description

`GetStart()`	• Called with two parameters:
	  ◦ a message string
	  ◦ an integer giving the number of the required hashtag. For example, `GetStart(Message, 3)` would search for the third hashtag in the string `Message`
	• Returns an integer value representing the start position of the hashtag in the message string, or value −1 if that hashtag does not exist

`GetTag()`	• Called with two parameters:
	  ◦ a message string
	  ◦ an integer giving the hashtag start position within the message
	• Returns the hashtag or an empty string if the character in the message at the hashtag start position is not `'#'`

`GetIndex()`	• Called with a hashtag as a parameter • Returns the index position of the hashtag in array `TagString` • Returns the value −1 if the hashtag is not present in the array

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

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

The module description is repeated here for reference.

Module	Description

`GetStart()`	• Called with two parameters:
	  ◦ a message string
	  ◦ an integer giving the number of the required hashtag. For example, `GetStart(Message, 3)` would search for the third hashtag in the string `Message`
	• Returns an integer value representing the start position of the hashtag in the message string, or value −1 if that hashtag does not exist

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

BLANK PAGE

(c) Write **program code** for the module `GetTag()`.

The module description is repeated here for reference.

Module	Description

`GetTag()`	• Called with two parameters:
	  ◦ a message string
	  ◦ an integer giving the hashtag start position within the message
	• Returns the hashtag or an empty string if the character in the message at the hashtag start position is not `'#'`

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]

Appendix

Built-in functions (pseudocode)

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

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

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

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

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

Example: `INT(27.5415)` returns `27`

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

`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 string `"BCD"`

`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'`

`DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`

Example: `DIV(10,3)` returns `3`

`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"`

© UCLES 2020

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`

© UCLES 2020

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.

© UCLES 2020

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

0996343222

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2016**

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/FD) 106688/3
© UCLES 2016

CAMBRIDGE International Examinations

There is an **Appendix** on page 18. Some questions will refer you to this information.

1 The items in the table below are individual statements in a generic programming language.

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

(a) (i) Show what type of programming construct each statement represents.

Complete the table by putting a tick (✓) in the appropriate column for each item.

Item	Statement	Selection	Iteration	Assignment
1	`MyScore = 65`			
2	`FOR IndexVal = 0 TO 99`			
3	`MyArray[3] = MID(MyString,3,2)`			
4	`IF MyScore >= 70 THEN`			
5	`ENDWHILE`			
6	`ELSE Message = "Error"`			

[6]

(ii) State the purpose of each statement in the table in **part (a)(i)**.

Do **not** use mathematical symbols in your descriptions.

Item	Purpose of statement
1	....
2	....
3	....
4	....
5	....
6	....

[6]

(iii) Evaluate the following expressions when `MyString` has the value `"Adaptive Maintenance"`.

Expression	Result
`'D' & RIGHT(MyString, 4)`	
`LEFT(RIGHT(MyString, 7), 3)`	

[2]

2 A team is designing a software system to monitor temperature in a process. To do this, the system needs to sample the temperature repeatedly. If the temperature exceeds a given threshold value, an alarm will sound.

The system is to be software-based. It will include a subroutine, `SampleTemp`, which samples the temperature and sets the alarm state to either ON or OFF.

The initial design stage will produce a prototype of `SampleTemp` with a user interface.
The structured English for this is:

1. IF the temperature does not exceed threshold value, SET alarm state to OFF
2. INPUT threshold value (to two decimal places)
3. INPUT sensor value (a whole number in the range 0 to 100)
4. MULTIPLY sensor value by conversion factor 1.135 to give temperature
5. IF temperature exceeds threshold value SET alarm state to ON
6. IF temperature exceeds threshold value OUTPUT message “Temperature Alarm”
7. IF temperature does not exceed threshold value OUTPUT message “Temperature OK”

(a) The procedure needs four variables. Complete the identifier table below for these variables.

Identifier	Data type	Description
`AlarmState`		....
`SensorValue`		....
`ThresholdValue`		....
`Temperature`		....

[4]

(b) Write the **pseudocode** equivalent of the structured English. Use the identifiers from the table in **part (a)**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

Question 3 begins on page 7.

3 A string encryption function is needed. The encryption uses a simple character-substitution method.

In this method, a new character substitutes for each character in the original string. This will create the encrypted string.

The substitution uses the 7-bit ASCII value for each character. This value is used as an index for a 1D array, `Lookup`, which contains the substitute characters.
`Lookup` contains an entry for each of the ASCII characters. It may be assumed that the original string and the substitute characters are all printable.

For example:

- 'A' has ASCII value 65
- Array element with index 65 contains the character 'Y' (the substitute character)
- Therefore, 'Y' substitutes for 'A'
- There is a different substitute character for every ASCII value

The programmer writes a function, `EncryptString`, to return the encrypted string. This function will receive two parameters, the original, `PlainText` string and the 1D array.

(a) The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode.

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

```
FUNCTION EncryptString(............................) RETURNS STRING

   DECLARE ..................... , ....................... : CHAR

   DECLARE OldCharValue : ....................................

   DECLARE n : INTEGER

   DECLARE OutString : STRING

   ............................ //initialise the return string

   //loop through PlainText to produce OutString

   FOR n ← 1 TO ................ //from first to last character

      OldChar ← ...................//get next character

      OldCharValue ← ..............//find the ASCII value

      NewChar ← ...................//look up substitute character

      ............................//concatenate to OutString

   ENDFOR

   ............................................................
ENDFUNCTION
```

[10]

(b) Additional code needs to be written to allow the user to change some of the characters in the array `Lookup`.

The user will input:

- the array start position
- the number of elements to change
- each new substitute character

At the end, the program will finally output a confirmation message.

The first version of the algorithm is represented by the flowchart on the following page.

(i) Write **program code** to declare the array `Lookup`.

Programming language .....

.....

.....

.....[2]

(ii) Write **program code** to implement the flowchart design.

In addition to the `Lookup` array, assume that the following variables have been declared:

```
StartPos, NumToChange, n, NewChar
```

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

START
Input StartPos
Input NumToChange
n ← 0
Loop
Output prompt for new substitute character
Input NewChar
Assign NewChar to current array element
n = NumToChange – 1?
No
Increment n
Yes
Output message stating number of elements changed
STOP

4 **(a)** Structured programming involves the breaking down of a problem into modules.

Give **two** reasons why this is done.

1 .....

.....

2 .....

.....

[2]

(b) A team needs to write a program to implement an online shopping system. Customers will access the program via a website.

Customers can search for items before adding them to a virtual shopping basket. When they have finished shopping, they pay for the items. The program provides output for the dispatch of the items.

Some of the key features of the system are as follows:

- a customer can add many items to the shopping basket
- payment may be either by credit or debit card, or by adding to a customer account
- the shop may dispatch the items in one or more packages

The structure chart below shows the program modules only.

(i) Draw on the chart, the symbols that represent the key features listed in **part (b)** above.

[3]

(ii) A section of the chart in **part (b)(i)** is shown below. It is to show the parameters passed between the Checkout and Card payment modules.

Name the three data items corresponding to the arrows.

Arrow	Data item
A	
B	
C	

[3]

Question 5 begins on page 13.

5 Toni has a large collection of jazz CDs that are stored in different places. She wants to record where the CDs are stored. She decides to write a program to do this.

The program must store the data in a file, `MyMusic`.

(a) **(i)** Why is a file needed?

.....

.....[1]

(ii) `MyMusic` is a text file with the data for each CD as one line of text.

Data for a typical CD are:

Title:	Kind of Green
Artist:	Miles Coltrane
Location:	Rack1-5

The line will be formed by concatenating the three data items.

For the example above, the line stored will be:

```
Kind of GreenMiles ColtraneRack1-5
```

Describe a problem that might occur when organising the data in this way.

.....

.....

.....

.....

Describe a possible solution.

.....

.....

.....

.....
[4]

(b) Toni must input the data into the file for all of her CDs.

A procedure, `InputData`, is needed to do this.

Toni designs the procedure and chooses the following identifiers:

Identifier	Data type
`CDTitle`	`STRING`
`CDArtist`	`STRING`
`CDLocation`	`STRING`

The procedure repeatedly performs the following steps:

- input a CD title (A rogue value of “##” is to be used to end the input)
- input the artist
- input the location
- create the text line
- write the text line to the file

When the rogue value is encountered the file is closed.

Write **program code** for the procedure `InputData`.

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

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

6 A string-handling function has been developed. The pseudocode for this function is shown below.

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

```
FUNCTION SSM(String1, String2 : STRING) RETURNS INTEGER
   DECLARE n, f, x, y : INTEGER

   n ← 0
   f ← 0

   REPEAT
      n ← n + 1
      x ← n
      y ← 1
      WHILE MID(String1, x, 1) = MID(String2, y, 1)

         IF y = LENGTH(String2)
            THEN
               f ← n
            ELSE
               x ← x + 1
               y ← y + 1
         ENDIF

      ENDWHILE

   UNTIL (n = LENGTH(String1)) OR (f <> 0)

   RETURN f

ENDFUNCTION
```

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

```
SSM("RETRACE", "RAC")
```

n	f	x	y	MID(String1, x, 1)	MID(String2, y, 1)
0	0				

[6]

(b) **(i)** Describe the purpose of function `SSM`.

.....

.....

.....

.....[2]

(ii) One of the possible return values from function `SSM` has a special meaning.

State the value and its meaning.

Value .....

Meaning .....
[2]

(iii) There is a problem with the logic of the pseudocode. This could generate a run-time error.

Describe the problem.

.....

.....

.....

.....[2]

Appendix

Built-in functions

In each function below, if the function call is not properly formed, the function returns an error.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`

returns the string of length `y` starting at position `x` from `ThisString`

Example: **`MID("ABCDEFGH", 2, 3)`** will return string **`"BCD"`**

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`

returns the leftmost `x` characters from `ThisString`

Example: **`LEFT("ABCDEFGH", 3)`** will return string **`"ABC"`**

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`

returns the rightmost `x` characters from `ThisString`

Example: **`RIGHT("ABCDEFGH", 3)`** will return string **`"FGH"`**

`ASC(ThisChar : CHAR) RETURNS INTEGER`

returns the ASCII value of character `ThisChar`

Example: **`ASC('W')`** will return **`87`**

`LENGTH(ThisString : STRING) RETURNS INTEGER`

returns the integer value representing the length of string `ThisString`

Example: **`LENGTH("Happy Days")`** will return **`10`**

String operator

`&` operator

concatenates (joins) two strings

Example: **`"Summer" & " " & "Pudding"`** produces **`"Summer Pudding"`**

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2016

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2022**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (NF/CB) 302729/4
© UCLES 2022

Refer to the **insert** for the list of pseudocode functions and operators.

1 A program is required for a shopping website.

(a) Part of the program requires four variables. The following table describes the use of each variable.

Complete the table by adding the most appropriate data type for each variable.

Variable use	Data type
Store the number of days in the current month	
Store the first letter of the customer's first name	
Store an indication of whether a year is a leap year	
Store the average amount spent per customer visit	

[4]

(b) The designer considers the use of a development life cycle to split the development of the website into several stages.

(i) State **one** benefit of a development life cycle when developing the website.

.....

..... [1]

(ii) Analysis is one stage of a development life cycle.

State **one** document that may be produced from the analysis stage of the website project.

.....

..... [1]

(c) The program will be developed using the Rapid Application Development (RAD) life cycle.

(i) State **one** principle of this life cycle.

.....

..... [1]

(ii) Give **two** benefits and **one** drawback of its use compared to the waterfall life cycle.

Benefit 1 .....

.....

Benefit 2 .....

.....

Drawback .....

.....
[3]

(d) Adaptive maintenance needs to be carried out on the website program.

Give **two** reasons why adaptive maintenance may be required.

1 .....

.....

2 .....

.....
[2]

2 A program is being designed for a smartphone to allow users to send money to the charity of their choice.

Decomposition will be used to break the problem down into sub-problems.

Identify **three** program modules that could be used in the design **and** describe their use.

Module 1 .....

Use .....

.....

.....

.....

Module 2 .....

Use .....

.....

.....

.....

Module 3 .....

Use .....

.....

.....

.....

[3]

3 An algorithm is needed to process a sequence of numbers. Numbers may be positive or negative and may be integer or decimal.

The algorithm will:

- prompt and input one number at a time until the value zero is input
- sum the negative numbers
- sum the positive numbers
- when zero is input, output the two sum values and then end.

Describe the algorithm needed. Do **not** include pseudocode statements in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

4 **(a)** A program contains a 1D array `DataItem` with 100 elements.

State the **one additional** piece of information required before the array can be declared.

.....

..... [1]

(b) A programmer decides to implement a queue Abstract Data Type (ADT) in order to store characters received from the keyboard. The queue will need to store at least 10 characters and will be implemented using an array.

(i) Describe **two** operations that are typically required when implementing a queue. State the check that must be carried out before each operation can be completed.

Operation 1 .....

.....

Check 1 .....

.....

Operation 2 .....

.....

Check 2 .....

.....
[4]

(ii) Describe the declaration and initialisation of the variables and data structures used to implement the queue.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

5 **(a)** A text string contains three data items concatenated as shown:

<StockID><Description><Cost>

Item lengths are:

Item	Length
StockID	5
Description	32
Cost	the remainder of the string

A procedure `Unpack()` takes four parameters of type string. One parameter is the original text string. The other three parameters are used to represent the three data items shown in the table and are assigned values within the procedure. These values will be used by the calling program after the procedure ends.

(i) Write pseudocode for the procedure `Unpack()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(ii) Explain the term **procedure interface** with reference to procedure `Unpack()`.

.....

.....

.....

..... [2]

(b) The design changes and a record structure is defined to store the three data items.

A user-defined data type `StockItem` is created as shown:

```
TYPE StockItem
   DECLARE StockID : STRING
   DECLARE Description : STRING
   DECLARE Cost : REAL
ENDTYPE
```

(i) A variable `LineData` of type `StockItem` is declared.

Write the pseudocode statement to assign the value `12.99` to the `Cost` field of `LineData`.

..... [1]

(ii) Procedure `Unpack()` is modified and converted to a function which takes the original text string as the only parameter.

Explain the other changes that need to be made to convert the procedure into a function.

.....

.....

.....

.....

..... [2]

(c) `Unpack()` is part of a program made up of several modules. During the design stage, it is important to follow good programming practice. One example of good practice is the use of meaningful identifier names.

Give the reason why this is good practice. Give **two other** examples of good practice.

Reason .....

.....

.....

Example 1 .....

.....

.....

Example 2 .....

.....

.....
[3]

(d) The program that includes `Unpack()` is tested using the walkthrough method.

Describe this method **and** explain how it can be used to identify an error.

.....

.....

.....

.....

.....

..... [3]

6 Components are weighed during manufacture. Weights are measured to the nearest whole gram.

Components that weigh at least 3 grams more than the maximum weight, or at least 3 grams less than the minimum weight, are rejected.

A component is rechecked if it weighs within 2 grams of either the maximum or minimum weight.

The final outcome of weighing each component is shown below:

A function `Status()` will be called with three parameters. These are integers representing the weight of an individual component together with the minimum and maximum weights.

The value returned from the function will be as follows:

Outcome	Return value
Accept	`'A'`
Reject	`'R'`
Recheck	`'C'`

(a) Complete the following test plan for **five** tests that could be performed on function `Status()`. The tests should address all possible outcomes.

Test number	Component weight	Min	Max	Expected return value
1				`'A'`
2				
3				
4				
5				

[5]

(b) Write pseudocode for `Status()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

7 A teacher is designing a program to perform simple syntax checks on programs written by students.

Two global 1D arrays are used to store the syntax error data. Both arrays contain 500 elements.

- Array `ErrCode` contains integer values that represent an error number in the range 1 to 800.
- Array `ErrText` contains string values that represent an error description.

The following diagram shows an example of the arrays.

Index	ErrCode	ErrText
1	10	"Invalid identifier name"
2	20	"Bracket mismatch"
3	50	""
4	60	"Type mismatch in assignment"
...		
500	999	<Undefined>

Note:

- There are less than 500 error codes so corresponding elements in both arrays may be unused. Unused elements in `ErrCode` have the value 999. These will occur at the end of the array. The value of unused elements in `ErrText` is undefined.
- Values in the `ErrCode` array are stored in ascending order but not all values may be present. For example, there may be no error code 31.
- Some error numbers are undefined. In these instances, the `ErrCode` array will contain a valid error number but the corresponding `ErrText` element will contain an empty string.

The teacher has defined one program module as follows:

Module	Description

`OutputRange()`	• Prompts for input of two error numbers
	• Outputs a list of error numbers between the two numbers input (inclusive) together with the corresponding error description
	• Outputs a warning message when the error description is missing as for error number 50 in the example
	• Outputs a suitable header and a final count of error numbers found
	
	Output based on the example array data above:
	
	`List of error numbers from 1 to 60`
	
	`10 : Invalid identifier name`
	`20 : Bracket mismatch`
	`50 : Error Text Missing`
	`60 : Type mismatch in assignment`
	
	`4 error numbers output`

(a) Write pseudocode for module `OutputRange()`. Assume that the two numbers input represent a valid error number range.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

Question 7 continues on the next page.

(b) (i) Two additional modules are defined:

Module	Description
`SortArrays()`	• Sorts the arrays into ascending order of `ErrCode`

`AddError()`	• Takes two parameters:
	  ○ an error number as an integer
	  ○ an error description as a string
	• Writes the error number and error description to the first unused element of the two arrays. Ensures the `ErrCode` array is still in ascending order
	• Returns the number of unused elements **after** the new error number has been added
	• Returns −1 if the new error number could not be added

Write pseudocode for the module `AddError()`. Assume that the error code is **not** already in the `ErrCode` array.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(ii) A new Module `RemoveError()` will remove a given error number from the array.

Describe the algorithm that would be required. Do **not** include pseudocode statements in your answer.

.....

.....

.....

.....

.....

.....

..... [3]

BLANK PAGE

BLANK PAGE

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 3 0 2 5 5 8 8 3 4 7 *

COMPUTER SCIENCE **9608/21**

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 **16** printed pages.

© UCLES 2019

Cambridge Assessment International Education

1 (a) (i) Algorithms may be expressed using four basic constructs. One construct is sequence.

Complete the following table for two other constructs.

Construct	Pseudocode example
.....	....
.....	....

[4]

(ii) Simple algorithms usually consist of input, process and output.

Complete the table by placing ticks ('✓') in the relevant boxes.

Pseudocode statement	Input	Process	Output
`Temp ← SensorValue * Factor`			
`WRITEFILE "LogFile.txt", TextLine`			
`WRITEFILE "LogFile.txt", MyName & MyIDNumber`			
`READFILE "AddressBook.txt", NextLine`			

[4]

(b) Program variables have values as follows:

Variable	Value
`Title`	`"101 tricks with spaghetti"`
`Version`	`'C'`
`Author`	`"Eric Peapod"`
`PackSize`	`4`
`WeightEach`	`6.2`
`Paperback`	`TRUE`

(i) 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 16.

Expression	Evaluates to
`MID(Title, 5, 3) & RIGHT(Author, 3)`	
`INT(WeightEach * PackSize)`	
`PackSize >= 4 AND WeightEach < 6.2`	
`LEFT(Author, ASC(Version) - 65)`	
`RIGHT(Title, (LENGTH(Author) - 6))`	

[5]

(ii) Programming languages support different data types.

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

Variable	Data type
`Title`	
`Version`	
`PackSize`	
`WeightEach`	
`Paperback`	

[5]

(c) White-box and black-box are two types of testing. In white-box testing, data are chosen to test every possible path through the program.

Explain how data are chosen in black-box testing.

.....

..... [2]

2 **(a)** One type of loop that may be found in an algorithm is a count-controlled loop.

State **one other** type **and** explain when it should be used.

Type .....

Explanation .....

.....

.....
[2]

(b) Chris is asked to work on a program that has been coded in a language he is not familiar with.

He has identified that the program contains the constructs: sequence, iteration and selection.

Identify **three other** features of the program that he should expect to recognise.

Feature 1 .....

Feature 2 .....

Feature 3 .....
[3]

(c) The following lines of code are taken from a program in a high-level language.

```
ON x {
      15: Call ProcA
      20: y := 0
      25: y := 99
      NONE: Call ProcError
     }
```

Identify the type of control structure **and** describe the function of the code.

Control structure .....

Description .....

.....

.....

.....
[3]

3 **(a)** A student is developing an algorithm to search through a 1D array of 100 elements. Each element of the array, `Result`, contains a `REAL` value.

The algorithm will output:

- the average value of all the elements
- the number of elements with a value of zero.

The structured English description of the algorithm is:

1. SET Total value to 0
2. SET Zero count to 0
3. SELECT the first element
4. ADD value of element to Total value
5. IF element value is 0 then INCREMENT Zero count
6. REPEAT from step 4 for next element, until element is last element
7. SET Average to Total / 100
8. OUTPUT a suitable message and Average
9. OUTPUT a suitable message and Zero count

Write **pseudocode** for this algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) The student decides to change the algorithm and implement it as a procedure, `ScanArray()`, which will be called with three parameters.

```
ScanArray(AverageValue, ZeroCount, ArrayName)
```

`ScanArray()` will modify the first two parameters so that the new values are available to the calling program or module.

Write the **pseudocode** procedure header for `ScanArray()`.

.....

.....

.....

..... [4]

Question 4 begins on the next page.

4 The following pseudocode is a string handling function.

For the built-in functions list, refer to the **Appendix** on page 16.

```
FUNCTION Clean(InString : STRING) RETURNS STRING

   DECLARE NewString : STRING
   DECLARE Index : INTEGER
   DECLARE AfterSpace : BOOLEAN
   DECLARE NextChar : CHAR
   CONSTANT Space = ' '

   AfterSpace ← FALSE
   NewString ← ""

   FOR Index ← 1 TO LENGTH(InString)
      NextChar ← MID(InString, Index, 1)
      IF AfterSpace = TRUE
         THEN
            IF NextChar <> Space
               THEN
                  NewString ← NewString & NextChar
                  AfterSpace ← FALSE
            ENDIF
         ELSE
            NewString ← NewString & NextChar
            IF NextChar = Space
               THEN
                  AfterSpace ← TRUE
            ENDIF
      ENDIF
   ENDFOR

   RETURN NewString

ENDFUNCTION
```

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

```
Result ← Clean("X∇∇∇Y∇and∇∇Z")
```

The symbol `'∇'` represents a space character. Use this symbol to represent a space character in the trace table.

Index	AfterSpace	NextChar	NewString

[6]

(ii) State the effect of the function `Clean()`.

.....

..... [1]

(iii) The pseudocode is changed so that the variable `AfterSpace` is initialised to `TRUE`.

Explain what will happen if the function is called as follows:

```
Result ← Clean("∇∇X∇∇∇Y∇and∇∇Z")
```

.....

.....

.....

..... [2]

(b) The following pseudocode declares and initialises an array.

```
DECLARE Code : ARRAY[1:100] OF STRING
DECLARE Index : INTEGER

FOR Index ← 1 TO 100
   Code[Index] ← ""
ENDFOR
```

The design of the program is changed as follows:

- the array needs to be two dimensional, with 500 rows and 4 columns
- the elements of the array need to be initialised to the string `"Empty"`

Re-write the **pseudocode** to implement the new design.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) State the term used for changes that are made to a program in response to a specification change.

..... [1]

Question 5 begins on the next page.

5 **(a)** Programming languages usually contain a range of built-in functions, such as a random number generator.

State **three** advantages of using built-in functions.

1 .....

2 .....

3 .....
[3]

(b) A student is learning about random number generation.

She is investigating how many times the random function needs to be called before every number in a given series is generated.

She is using **pseudocode** to develop a procedure, `TestRand()`, which will:

- use the random number function to generate an integer value in the range 1 to 50 inclusive
- count how many times the random function needs to be called before all 50 values have been generated
- output a message giving the number of times the random function was called.

Write **pseudocode** for the procedure `TestRand()`.

For the built-in functions list, refer to the **Appendix** on page 16.

..... [8]

6 A text file, `MyCDs.txt`, stores information relating to a Compact Disc (CD) collection. Information about each CD is stored on three separate lines in the file as follows:

```
Line 1:   <Artist Name>
Line 2:   <CD Title>
Line 3:   <Storage Location>
```

Information is stored as data strings.

A section of the file is shown:

File line	Data
100	`"Green Floyd"`
101	`"Bowlful of Cereal"`
102	`"Shelf 4"`
103	`"Strolling Bones"`
104	`"Exile on Station Road"`
105	`"Box 12"`

(a) A program, `CDOrganiser`, will be written to manage the stored information. The program will consist of three modules: `AddCD`, `FindCD` and `RemoveCD`.

Give **three** reasons why it is good practice to construct the program using modules.

1 .....

2 .....

3 .....
[3]

(b) The module, `FindCD()`, will check whether a given CD exists in the collection. The module will be implemented as a function.

The function will:

- be called with two strings as parameters, representing the artist name and CD title
- return a string that gives the storage location, or an empty string if the given CD has not been found.

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.

© UCLES 2019

Write **program code** for the function `FindCD()`.

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]

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`

`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` (`x` not inclusive).

Example: `RAND(87)` could return `35.43`

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`

© UCLES 2019

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 2020**

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 **20** pages. Blank pages are indicated.

DC (RW/TP) 180778/4
© UCLES 2020

BLANK PAGE

1 **(a)** Algorithms are produced during program development.

State when you would produce an algorithm during program development **and** state its purpose.

When .....

.....

Purpose .....

.....
[2]

(b) Selection is one of the basic constructs used in algorithms.

Explain the term **selection**.

.....

.....

.....

..... [2]

(c) Explain the process of **problem decomposition**. State one reason it may be used.

Explanation .....

.....

.....

Reason .....

.....

.....
[2]

(d) Name **two** features provided by a typical Integrated Development Environment (IDE) that assist in the **debugging** stage of the program development cycle.

1 .....

2 .....
[2]

2 **(a)** A structure chart is often produced as part of a modular program design. The chart shows the relationship between modules and the parameters that are passed between them.

Give **two other** features the structure chart can show.

Feature 1 .....

.....

Feature 2 .....

.....
[2]

(b) The following structure chart shows the relationship between three modules.

Parameter data types are:

```
ParW : REAL
ParX : INTEGER
ParZ : STRING
```

(i) Write the **pseudocode** header for module `ModuleB()`.

.....

.....

..... [3]

(ii) Write the **pseudocode** header for module `ModuleC()`.

.....

.....

..... [3]

(c) A student is developing an algorithm to count the number of times a given string, `SearchString`, appears in a 2D array. The array, `Item`, consists of 100 elements organised as 50 rows of 2 columns. `SearchString` could appear in any row or column.

The array is declared in pseudocode as follows:

```
DECLARE Item : ARRAY [1:50, 1:2] OF STRING
```

The structured English description of the algorithm is:

1. SET Count to 0.
2. Examine the first row of the array.
3. IF column 1 element value is equal to `SearchString`, ADD 1 to Count.
4. IF column 2 element value is equal to `SearchString`, ADD 1 to Count.
5. REPEAT from step 3 for next row, UNTIL row is last row.
6. OUTPUT a suitable message and Count.

Write **pseudocode** for the algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

BLANK PAGE

© UCLES 2020

3 **(a)** Study the following pseudocode.

```
Error ← Delta(Plan, Actual)
CASE OF Error
    > 10 : Steer ← Steer - 10
       0 : ZCount ← ZCount + 1
   < -10 : Steer ← Steer + 10
   OTHERWISE OUTPUT "Unexpected Error"
ENDCASE
```

Draw a program flowchart to represent the pseudocode. Variable declarations are **not** required in program flowcharts.

[5]

(b) The following pseudocode algorithm has been developed to check whether a string contains a valid password.

To be a valid password, a string must:

- be longer than five characters
- contain at least one numeric digit
- contain at least one upper case letter
- contain at least one other character (not a numeric digit or an upper case letter).

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

  DECLARE Index : INTEGER
  DECLARE StrLen : INTEGER
  DECLARE NumUpper, NumDigit : INTEGER
  DECLARE NextChar : CHAR
  DECLARE NumOther : INTEGER

  NumUpper ← 0
  NumDigit ← 0

  StrLen ← LENGTH(InString)
  IF StrLen < 6
     THEN
        RETURN FALSE
     ELSE
        FOR Index ← 1 TO StrLen - 1
           // loop for each character
           NextChar ← MID(InString, Index, 1)
           IF NextChar >= '0' AND NextChar <= '9'
              THEN
                 NumDigit ← NumDigit + 1 // count digits
              ELSE
                 IF NextChar >= 'A' AND NextChar <= 'Z'
                    THEN
                       NumUpper ← NumUpper + 1 // count upper case
                 ENDIF
           ENDIF
        ENDFOR
  ENDIF

  NumOther ← StrLen – (NumDigit – NumUpper)
  IF NumDigit >= 1 AND NumUpper >= 1 AND NumOther >= 1
     THEN
        RETURN TRUE
     ELSE
        RETURN FALSE
  ENDIF

ENDFUNCTION
```

The pseudocode does not work under all circumstances.

The function was dry run with the string "1234AP" and the following trace table was produced. The string is an invalid password, but the pseudocode returned the value TRUE.

Trace table row	StrLen	Index	NextChar	NumUpper	NumDigit	NumOther
1	6			0	0	
2		1	'1'			
3					1	
4		2	'2'			
5					2	
6		3	'3'			
7					3	
8		4	'4'			
9					4	
10		5	'A'			
11				1		
12						3

(i) The pseudocode algorithm contains two errors.

State how the given trace table indicates the existence of each error.

Error 1 .....

.....

.....

.....

Error 2 .....

.....

.....

.....

[2]

(ii) Give the line number of each error in the pseudocode algorithm **and** write the modified pseudocode to correct each error.

Line number for error 1 .....

Correct pseudocode .....

.....

Line number for error 2 .....

Correct pseudocode .....

.....
[2]

(c) The term **adaptive maintenance** refers to amendments that are made in response to changes to the program specification. These changes usually affect the program algorithm.

Name **one other** part of the design that can change as a result of adaptive maintenance.

..... [1]

4 A global 1D array, `Contact`, of type `STRING` is used to store a list of names and email addresses. There are 1000 elements in the array. Each element stores one data item. The format of each data item is as follows:

```
<Name>':'<EmailAddress>
```

`Name` and `EmailAddress` are both variable-length strings.

For example:

```
"Wan Zhu:zwan99@mymail.com"
```

A function, `Extract()`, is part of the program that processes the array. A string data item is passed to the function as a parameter. The function will return the `Name` part. Validation is **not** necessary.

(a) Write **program code** for the function `Extract()`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) The original function, `Extract()`, needs to be modified to separate the name from the email address. The calling program can then use **both** of these values.

Write, in **pseudocode**, the header for the modified subroutine. Explain the changes you have made.

Subroutine header .....

.....

Explanation .....

.....

.....

.....

.....

.....
[3]

5 A company hires out rowing boats on a lake. The company has 20 boats, numbered from 1 to 20.

For safety reasons, the boats have to be serviced (checked and any damage repaired) regularly.

The company is developing a program to help manage the servicing of the boats.

Every time a boat is serviced, details are added at the end of the text file, `ServiceLog.txt`, as a single line of information. Each boat is serviced before it is hired out for the first time.

The format of each line is as follows:

```
<BoatNumber><Date>
```

`BoatNumber` and `Date` are as follows:

- `BoatNumber` is a two-digit numeric string in the range `"01"` to `"20"`
- `Date` is an 8-digit numeric string in the format YYYYMMDD

The programmer has defined the first module as follows:

Module	Description
`GetLastService()`	• Called with a string parameter representing the `BoatNumber` • Searches through the file `ServiceLog.txt` • Returns the date of the last service in the form "YYYYMMDD"

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

Refer to the **Appendix** on page 19 for a list of built-in pseudocode functions and operators.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(b) (i) Every time a boat is hired out, details of the hire are added at the end of a text file, `Hirelog.txt`. Each line of the text file corresponds to information about the hire of one boat.

The format of each line of information is as follows:

```
<Date><BoatNumber><HireDuration>
```

- `Date` is an 8-digit numeric string in the format YYYYMMDD
- `BoatNumber` is a two-digit numeric string in the range `"01"` to `"20"`
- `HireDuration` is a variable-length string representing a numeric value in hours. For example, the string `"1.5"` would represent a hire duration of 1½ hours.

A module `GetHours()` is defined as follows:

Module	Description
`GetHours()`	• Takes two parameters: ◦ the `BoatNumber` as a string ◦ the date of the last service for that boat ("YYYYMMDD") as a string • Searches through file `Hirelog.txt` and calculates the sum of the hire durations for the given boat **after** the given date (hire durations **on or before** the given date are ignored) • Returns the total of the hire durations as a real

Note:

Standard comparison operators may be used with dates in this format.
For example:

`"20200813" > "20200812"` would evaluate to `TRUE`

Parameter validation is **not** required.

Write **pseudocode** for the module `GetHours()`.

Refer to the **Appendix** on page 19 for a list of built-in pseudocode functions and operators.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(ii) An additional module, `Validate()`, has been written to check that a given string corresponds to a valid `BoatNumber`. A valid `BoatNumber` is a two-digit numeric string in the range `"01"` to `"20"`.

Give **three** test strings that are **invalid** for different reasons. Explain your choice in each case.

String 1 .....

Reason .....

.....

.....

String 2 .....

Reason .....

.....

.....

String 3 .....

Reason .....

.....

.....

[6]

(c) A new module is described as follows:

Module	Description

`ServiceList()`	• Takes an integer as a parameter that represents the maximum number of hours before a boat must be serviced
	• Uses `GetLastService()` and `GetHours()`
	• Outputs:
	◦ a suitable heading
	◦ the `BoatNumber` of each boat hired for more than the maximum number of hours since its last service
	◦ the total hire duration for each of these boats
	
	An example output list is:
	`Boat Service List`
	`4: 123`
	`17: 117`
	
	If no boats are due to be serviced, the output is:
	`Boat Service List`
	`No boats are due to be serviced`

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

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

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(d) **(i)** A team of programmers will work on the program. Before they begin, the team meet to discuss ways in which the risk of program faults may be reduced during the design and coding stages.

State **two** ways to minimise program faults during the design and coding stages.

1 .....

.....

2 .....

.....
[2]

(ii) During development, the team test the program using a process known as stub testing.

Explain this process.

.....

.....

.....

..... [2]

(iii) Explain how single stepping may be used to help find a logic error in a program.

.....

.....

.....

..... [2]

Appendix

Built-in functions (pseudocode)

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

`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"`

`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"`

`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`

© UCLES 2020

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.

© UCLES 2020

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2022**

INSERT **2 hours**

7344032364-I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (PQ) 302744/3
© UCLES 2022

Functions

Note: an error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

`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"`
`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`
`LCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the lower-case equivalent of `ThisChar` Non upper-case alphabetic characters are returned unchanged. Example: `LCASE('W')` returns `'w'`
`UCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the upper-case equivalent of `ThisChar` Non lower-case alphabetic characters are returned unchanged. Example: `UCASE('a')` returns `'A'`
`TO_UPPER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to upper case. Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`
`TO_LOWER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to lower case. Example: `TO_LOWER("JIM 803")` returns `"jim 803"`
`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>` returns a string representation of a numeric value. Note: `<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING` Example: `NUM_TO_STR(87.5)` returns `"87.5"`
`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>` returns a numeric representation of a string. Note: `<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER` Example: `STR_TO_NUM("23.45")` returns `23.45`

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN` returns the value `TRUE` if `ThisString` represents a valid numeric value. Note: `<datatype>` may be `CHAR` or `STRING` Example: `IS_NUM("-12.36")` returns `TRUE`
`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66` etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'` etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`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`

Date Functions

Note: date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2 etc. Example: `DAYINDEX(12/05/2020)` returns `3`
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a variable of type `DATE` with the value of `<Day>/<Month>/<Year>` Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26 October 2003
`TODAY() RETURNS DATE` returns a variable of type `DATE` corresponding to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: the function will generate an error if the file is not already open in `READ` mode.

Operators

Note: an error will be generated if an operator is used with a value or values of an incorrect type.

`&`	concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
 	
`>`
 	
`<`
 	
`>=`
 	
`<=`
 	
`<>`
	used to compare two items of the same type
	returns `TRUE` if the condition is true, otherwise returns `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when variable `Count` contains the value `4`

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2022**

INSERT **2 hours**

0832918086–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (JP) 311805
© UCLES 2022

Functions

Note: An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

`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"`
`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 string `"BCD"`
`LENGTH(ThisString : STRING) RETURNS INTEGER` returns the integer value representing the length of string `ThisString` Example: `LENGTH("Happy Days")` returns `10`
`LCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the lower-case equivalent of `ThisChar` Alphabetic characters that are not upper case are returned unchanged. Example: `LCASE('W')` returns `'w'`
`UCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the upper-case equivalent of `ThisChar` Alphabetic characters that are not lower case are returned unchanged. Example: `UCASE('a')` returns `'A'`
`TO_UPPER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to upper case. Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`
`TO_LOWER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to lower case. Example: `TO_LOWER("JIM 803")` returns `"jim 803"`
`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>` returns a string representation of a numeric value. Note: `<datatype1>` may be `REAL` or `INTEGER`, <datatype2> may be `CHAR` or `STRING` Example: `NUM_TO_STR(87.5)` returns `"87.5"`
`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>` returns a numeric representation of a string. Note: `<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER` Example: `STR_TO_NUM("23.45")` returns `23.45`

String and Character Functions

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN` returns the value `TRUE` if `ThisString` represents a valid numeric value. Note: `<datatype>` may be `CHAR` or `STRING` Example: `IS_NUM("-12.36")` returns `TRUE`
`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`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`

Date Functions

Note: Date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2, etc. Example: `DAYINDEX(12/05/2020)` returns `3`
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a variable of type `DATE` with the value of `<Day>/<Month>/<Year>`
`TODAY() RETURNS DATE` returns a variable of type `DATE` with the value set to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: The function will generate an error if the file is not already open in `READ` mode.

Operators

Note: An error will be generated if an operator is used with a value or values of an incorrect type.

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: May also be used to concatenate a `CHAR` with a `STRING`
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	Used to compare two items of the same type. Returns `TRUE` if the condition is true, otherwise returns `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records.
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `count = 4` evaluates to `TRUE` when variable `count` contains the value `4`

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

3659076713

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
© UCLES 2021

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`.

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

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

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.

© UCLES 2021

Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

1209867308

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.

DC (ST) 171121/3
© UCLES 2019

Cambridge Assessment International Education

[Turn over

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

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
`VolLevel`	The current volume setting. This must be in the range 0 to 49.
`MaxVol`	A value that can be set to limit the maximum value of `VolLevel`, 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.

(a) Write **pseudocode** for the procedure `Button()`. Declare any additional variables you use.

The value of `MaxVol` should **not** be changed within the procedure.

Parameter validation is **not** necessary.

..... [6]

(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]

© UCLES 2019

(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

`ProcessStudentList()`	• Read each line from the file `StudentList.txt`
	  ◦ Check whether the `Reference` appears in the array using `SearchLeavers()`
	  ◦ If the `Reference` does **not** appear then write the line to the file `UpdatedList.txt`
	• Return the number of lines **not** copied.

`SearchLeavers()`	• Search for a given `Reference` in the array `Leavers` • If the `Reference` is found, return `TRUE`, otherwise return `FALSE`
`CountTimes()`	• 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.

(a) Write **program code** for the module `SearchLeavers()`. Declare any additional variables you use.

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]

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

The module description is repeated here for reference.

Module	Outline description

`ProcessStudentList()`	• Read each line from the file `StudentList.txt`
	  ◦ Check whether the `Reference` appears in the array using `SearchLeavers()`
	  ◦ If the `Reference` does **not** appear then write the line to the file `UpdatedList.txt`
	• 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
`CountTimes()`	• 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`

© UCLES 2019

BLANK PAGE

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.

© UCLES 2019

Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

9534481872

COMPUTER SCIENCE **9608/23**

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 **16** printed pages.

DC (ST/SG) 171122/4
© UCLES 2019

Cambridge Assessment International Education

[Turn over

Question 1 begins on the next 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.

© UCLES 2019

1 **(a)** **(i)** Programming languages can support different data types.

Complete the table by naming **three** different data types together with an example data value for each.

Data type	Example data value

[6]

(ii) Identify the type of programming statement that assigns a data type to a variable.

..... [1]

(b) As part of the development of an algorithm, a programmer may construct an identifier table.

Describe what an identifier table contains.

.....

.....

.....

..... [2]

(c) **(i)** Simple algorithms usually consist of three different stages.

Complete the table below. Write each example statement in **program code**.

The second stage has already been given.

Stage	Example statement
Process	

[5]

(ii) Write a **single** statement in **program code** that contains **two** of the stages. Do **not** repeat any of the statements from **part (c)(i)**.

..... [1]

(d) A software developer is writing a program and includes several features to make it easier to read and understand. One of these features is the use of indentation.

State **three other** features.

Feature 1 .....

Feature 2 .....

Feature 3 .....
[3]

(e) A trace table is often used during program testing.

Identify the type of testing that includes the use of a trace table.

..... [1]

2 **(a)** **(i)** Two types of loop that may be found in an algorithm are the 'pre-condition' and 'post-condition' loop.

Identify **one other** type of loop. Explain when it should be used.

Type .....

Explanation .....

.....

.....
[2]

(ii) Part of a program flowchart is shown.

Implement the flowchart in **pseudocode** using a post-condition loop.

.....

.....

.....

.....

.....

..... [4]

(b) The following lines of code are taken from a high-level language program.

```
100 setvar(Count, Integer)
110 setvar(Gross[0-20], Real)
120 setvar(Posn, Real)
130 setvar(Length, Integer)
140 setvar(Rate, Real)
150 Length := 7
160 Rate := 1.175
170
180 For (Count, 0, 20, 2)
190  {
200    Echo "Input next cost"
210    Posn := Read()
220    Gross[Count] := Mult(Posn, Rate) %Apply current tax rate
230  }
```

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

Feature	Answer
The symbol used to indicate an assignment	
The line numbers for the start and end of a count-controlled loop	
The step value of the count-controlled loop	
The character that indicates a comment	
The name of a function	

[5]

(c) A program written in a high-level language cannot be run directly.

Identify **one** type of translator that can be used to translate the program.

..... [1]

3 Three program modules process updating of passwords in a file. A description of the relationship between the modules is summarised as follows:

Module name	Description
`GetPassword()`	• Takes two parameters: `AccountID` and `OldPassword` • Returns a string containing the new password
`UpdateFile()`	• Takes two parameters: `AccountID` and `NewPassword` • Returns a Boolean value to indicate whether or not the update was successful
`ChangePassword()`	• Calls `GetPassword()` to obtain the new password then calls `UpdateFile()` to write the new password to the file

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

[5]

4 The following pseudocode algorithm checks whether a string is a valid email address.

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

   DECLARE Index : INTEGER
   DECLARE NumDots : INTEGER
   DECLARE NumAts : INTEGER
   DECLARE NextChar : CHAR
   DECLARE NumOthers : INTEGER

   NumDots ← 0
   NumAts ← 0
   NumOthers ← 0

   FOR Index ← 1 TO LENGTH(InString)

      NextChar ← MID(InString, Index, 1)
      CASE OF NextChar
         '.': NumDots ← NumDots + 1
         '@': NumAts ← NumAts + 1
         OTHERWISE NumOthers ← NumOthers + 1
      ENDCASE

   ENDFOR

   IF (NumDots >= 1 AND NumAts = 1 AND NumOthers > 5)
      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 ← Check("Jim.99@skail.com")
```

Index	NextChar	NumDots	NumAts	NumOthers

[5]

(ii) State the value returned when function `Check` is called as shown in **part (b)(i)**.

..... [1]

(c) The function `Check()` is to be tested.

State **two** different invalid string values that could be used to test the algorithm. Each string should test a different rule.

Justify your choices.

Value .....

Justification .....

.....

.....

Value .....

Justification .....

.....

.....

[4]

5 Abbreviations are often used in place of a full name. Concatenating the first letter of each word in the name makes an abbreviation.

For example:

Name	Abbreviation
United Nations	UN
World Wide Web	WWW
British Computer Society	BCS

A function, `Abbreviate()`, will take a string representing the full name and return a string containing the abbreviated form.

You should assume that:

- names only contain alphabetic characters and space characters
- names always start with an alphabetic character
- each word in the name always starts with an uppercase character
- only a single space separates words in the name.

Write **pseudocode** to implement the function `Abbreviate()`.

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

..... [8]

6 A text file, `Library.txt`, stores information relating to a book collection. The file stores four pieces of information about each book on separate lines of the file, as follows:

```
Line n:      <Book Title>
Line n + 1:  <Author Name>
Line n + 2:  <ISBN>
Line n + 3:  <Location>
```

Information is stored as data strings.

Information relating to two books is shown:

File line	Data
100	"Learning Python"
101	"Brian Smith"
102	"978-14-56543-21-8"
103	"BD345"
104	"Surviving in the mountains"
105	"C T Snow"
106	"978-35-17635-43-9"
107	"ZX001"

(a) (i) A function, `FindBooksBy()`, will search `Library.txt` for all books by a given author.

The function will store the `Book Title` and `Location` in the array `Result`, and will return a count of the number of books found.

Array `Result` is a global 2D array of type `STRING`. It has 100 rows and 2 columns.

Write **pseudocode** to declare the array `Result`.

.....

.....

..... [3]

(ii) Function `FindBooksBy()` will:

- receive the `Author Name` as a parameter
- search `Library.txt` for matching entries
- store the `Book Title` and `Location` of matching entries in the `Result` array
- return an integer value giving the number of books by the author that were found.

Write **program code** for the function `FindBooksBy()`.

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) The function `FindBooksBy()` has already been called and has stored values in the array `Result`.

The procedure, `DisplayResults()`, will output the information from the array.

The procedure receives the following two parameters:

- a string containing the author name
- an integer value representing the number of books found

The output should be formatted as in the following example:

```
Books written by: Brian Smith

Title                     Location
Learning Python           BD345
Arrays are not lists      CZ562
Learning Java             CZ589

Number of titles found: 3
```

If no books by the author are found, the following should be output:

```
Search found no books by: Brian Smith
```

Write **pseudocode** for the procedure `DisplayResults()`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [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`

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

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

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

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`

© UCLES 2019

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

7483199901

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2021**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

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

DC (PQ/FC) 200587/4
© UCLES 2021

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** A program is being developed to help manage the membership of a football club.

Complete the following identifier table.

Example value	Explanation	Variable name	Data type
`"Wong"`	The preferred name of the member joining the football club		
`FALSE`	A value to indicate whether an existing member of the club lives at the same address		
`19/02/1983`	When the member joined the football club		
`1345`	The number of points a member has earned. Members of the club earn points for different activities.		

[4]

(b) 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.

You can assume that none of the variables referenced are of an incorrect type.

Statement	Error
`Result ← 2 & 4`	
`SubString ← MID("pseudocode", 4, 1)`	
`IF x = 3 OR 4 THEN`	
`Result ← Status AND INT(x/2)`	
`Message ← "Done" + LENGTH(MyString)`	

[5]

(c) The following data items need to be stored for each student in a group:

- student name (a string)
- test score (an integer).

State a suitable data structure and justify your answer.

Structure .....

Justification .....

.....

.....
[3]

2 **(a)** Four program modules form part of a program for a library.

A description of the relationship between the modules is summarised as follows:

Module name	Description
`UpdateLoan()`	• Calls either `LoanExtend()` or `LoanReturn()`
`LoanExtend()`	• Called with parameters `LoanID` and `BookID` • Calls `CheckReserve()` to see whether the book has been reserved for another library user • Returns `TRUE` if the loan has been extended, otherwise returns `FALSE`
`CheckReserve()`	• Called with `BookID` • Returns `TRUE` if the book has been reserved, otherwise returns `FALSE`
`LoanReturn()`	• Called with parameters `LoanID` and `BookID` • Returns a `REAL` (which is the value of the fine to be paid in the case of an overdue loan)

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

[5]

(b) The definition for module `LoanReturn()` is amended as follows:

Module name	Description
`LoanReturn()`	Called with parameters `LoanID`, `BookID` and `Fine` The module code checks whether the book has been returned on time and then assigns a new value to `Fine`

- `LoanID` and `BookID` are of type `STRING`
- `Fine` is of type `REAL`

Write the pseudocode header for the **amended** module `LoanReturn()`.

.....

..... [2]

(c) A program will:

- input 50 unique integer values
- output the largest value
- output the average of the values **excluding** the largest value.

Draw a program flowchart to represent the algorithm.

Variable declarations are **not** required.

It is not necessary to check that each input value is unique.

[6]

BLANK PAGE

3 **(a)** A concert venue uses a program to calculate admission prices and store information about ticket sales.

A number of arrays are used to store data. The computer is switched off overnight and data has to be input again at the start of each day before any tickets can be sold. This process is very time consuming.

(i) Explain how the program could use text files to speed up the process.

.....

.....

.....

.....

.....

..... [2]

(ii) State the characteristic of text files that allow them to be used as explained in **part (a)(i)**.

.....

..... [1]

(iii) Information about ticket sales will be stored as a booking. The booking requires the following data:

- name of person booking
- number of people in the group (for example a family ticket or a school party)
- event type.

Suggest how data relating to each booking may be stored in a text file.

.....

.....

.....

..... [2]

(b) A procedure `Preview()` will:

- take the name of a text file as a parameter
- output a warning message if the file is empty
- otherwise output the first five lines from the file (or as many lines as there are in the file if this number is less than five).

Write pseudocode for the procedure `Preview()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

4 Study the following pseudocode. Line numbers are for reference only.

```
10 FUNCTION Convert(Name : STRING) RETURNS STRING
11
12   DECLARE Flag: BOOLEAN
13   DECLARE Index : INTEGER
14   DECLARE ThisChar : CHAR
15   DECLARE NewName : STRING
16
17   CONSTANT SPACECHAR = ' '
18
19   Flag ← TRUE
20   Index ← 1
21   NewName ← ""      // formatted name string
22
23   WHILE Index <= LENGTH(Name)
24      ThisChar ← MID(Name, Index, 1)
25      IF Flag = TRUE THEN
26         NewName ← NewName & UCASE(ThisChar)
27         IF ThisChar <> SPACECHAR THEN
28            Flag ← FALSE
29         ENDIF
30      ELSE
31         NewName ← NewName & ThisChar
32      ENDIF
33      IF ThisChar = SPACECHAR THEN
34         Flag ← TRUE
35      ENDIF
36      Index ← Index + 1
37   ENDWHILE
38
39   RETURN NewName
40
41 ENDFUNCTION
```

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

```
Result ← Convert("∇in∇a∇∇Cup")
```

Note: The symbol `'∇'` has been used to represent a space character.
Use this symbol for any space characters in the trace table.

The first row has been completed for you.

Name	Flag	Index	NewName	ThisChar
"∇in∇a∇∇Cup"				

[5]

(b) The pseudocode for `Convert()` contains a conditional loop.

State a more appropriate loop structure.

Justify your answer.

Loop structure .....

.....

Justification .....

.....

.....
[2]

(c) Two changes need to be made to the algorithm.

Change 1: Convert to lower case any character that is not the first character after a space.
Change 2: Replace multiple spaces with a single space.

(i) Change 1 may be implemented by modifying one line of the pseudocode.

Write the modified line.

.....

.....[1]

(ii) Change 2 may be implemented by moving one line of the pseudocode.

Write the number of the line to be moved and state its new position.

Line number .....

New position .....

.....
[2]

BLANK PAGE

5 A global 2D array `Result` of type `INTEGER` is used to store a list of exam candidate numbers together with their marks. The array contains 2000 elements, organised as 1000 rows and 2 columns.

Column 1 contains the candidate number and column 2 contains the mark for the corresponding candidate. All elements contain valid exam result data.

A procedure `Sort()` is needed to sort `Result` into ascending order of mark using an efficient bubble sort algorithm.

Write pseudocode for the procedure `Sort()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

6 The following diagram represents an Abstract Data Type (ADT) for a linked list.

The free list is as follows:

(a) Explain how a node containing data value B is added to the list in alphabetic sequence.

.....

.....

.....

.....

.....

.....

..... [4]

(b) Describe how the linked list in **part (a)** may be implemented using variables and arrays.

.....

.....

.....

..... [2]

BLANK PAGE

7 A program is needed to take a string containing a full name and produce a new string of initials.

Some words in the full name will be ignored. For example, "the", "and", "of", "for" and "to" may all be ignored.

Each letter of the abbreviated string must be upper case.

For example:

Full name	Initials
Integrated Development Environment	IDE
The American Standard Code for Information Interchange	ASCII

The programmer has decided to use a global variable `FNString` of type `STRING` to store the full name.

It is assumed that:

- words in the full name string are separated by a single space character
- space characters will not occur at the beginning or the end of the full name string
- the full name string contains at least one word.

The programmer has started to define program modules as follows:

Module	Description
`GetStart()`	• Called with an `INTEGER` as a parameter, representing the number of a word in `FNString`. • Returns the character start position of that word in `FNString` or returns `-1` if that word does not exist • For example: if `FNString` contains the string `"hot and cold"`, `GetStart(3)` returns `9`
`GetWord()`	• Called with a parameter representing the position of the first character of a word in `FNString` • Returns the word from `FNString` • For example: if `FNString` contains the string `"hot and cold"`, `GetWord(9)` returns `"cold"`

(a) Write pseudocode for the module `GetStart()`.

.....

.....

.....

.....

.....

.....

..... [7]

(b) The programmer has decided to use a global ten-element 1D array `IgnoreList` of type `STRING` to store the ignored words. Unused elements contain the empty string (`""`) and may occur anywhere in the array.

A new module `AddWord()` is needed as follows:

Module	Description
`AddWord()`	• Called with a parameter representing a word • Stores the word in an unused element of the `IgnoreList` array and returns `TRUE` • Returns `FALSE` if the array was already full or if the word was already in the array

Write a detailed description of the algorithm for `AddWord()`. Do **not** include pseudocode statements in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) As a reminder, the module description of `GetWord()` is repeated:

Module	Description
`GetWord()`	• Called with a parameter representing the position of the first character of a word in `FNString` • Returns the word from `FNString` • For example: if `FNString` contains the string `"hot and cold"`, `GetWord(9)` returns `"cold"`

Write pseudocode for the module `GetWord()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

BLANK PAGE

BLANK PAGE

© UCLES 2021

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2023**

INSERT **2 hours**

3141164855-I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (DE) 328288
© UCLES 2023

An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

- A string of length 1 may be either of type `CHAR` or `STRING`
- A `CHAR` may be assigned to, or concatenated with, a `STRING`
- A `STRING` of length greater than 1 cannot be assigned to a `CHAR`

`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"`
`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 string `"BCD"`
`LENGTH(ThisString : STRING) RETURNS INTEGER` returns the integer value representing the length of `ThisString` Example: `LENGTH("Happy Days")` returns `10`

`TO_UPPER(x : <datatype>) RETURNS <datatype>`
`<datatype>` may be `CHAR` or `STRING`
 	
returns an object of type `<datatype>` formed by converting all characters of `x` to upper case.
 	
Examples:
• `TO_UPPER("Error 803")` returns `"ERROR 803"`
• `TO_UPPER('a')` returns `'A'`

`TO_LOWER(x : <datatype>) RETURNS <datatype>`
`<datatype>` may be `CHAR` or `STRING`
 	
returns an object of type `<datatype>` formed by converting all characters of `x` to lower case.
 	
Examples:
• `TO_LOWER("JIM 803")` returns `"jim 803"`
• `TO_LOWER('W')` returns `'w'`

`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>`
 	
returns a string representation of a numeric value.
`<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING`
Example: `NUM_TO_STR(87.5)` returns `"87.5"`

`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>`
 	
returns a numeric representation of a string.
`<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER`
Example: `STR_TO_NUM("23.45")` returns `23.45`

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN`
 	
returns `TRUE` if `ThisString` represents a valid numeric value.
`<datatype>` may be `CHAR` or `STRING`
Example: `IS_NUM("-12.36")` returns `TRUE`

`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`RAND(x : INTEGER) RETURNS REAL` returns a real number in the range `0` to `x` (**not** inclusive of `x`). Example: `RAND(87)` may return `35.43`

Date Functions

Date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the day index number from `ThisDate` where Sunday = 1, Monday = 2 etc. Example: `DAYINDEX(09/05/2023)` returns 3
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a value of type `DATE` with the value of `<Day>/<Month>/<Year>` Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26/10/2003
`TODAY() RETURNS DATE` returns a value of type `DATE` corresponding to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` The function will generate an error if the file is not already open in `READ` mode.

Operators

An error will be generated if an operator is used with a value or values of an incorrect type.

`&`	concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` The operator may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values. Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values. Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value. Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another. Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another. Example `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	used to compare two items of the same type.
	evaluates to `TRUE` if the condition is true, otherwise evaluates to `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when variable `Count` contains the value 4

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

1790695172

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2023**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages.

DC (LK/CT) 315870/2
© UCLES 2023

Refer to the **insert** for the list of pseudocode functions and operators.

1 A program is being developed in pseudocode before being converted into a programming language.

(a) The following table shows four valid pseudocode assignment statements.

Complete the table by giving the data type that should be used to declare the variable underlined in each assignment statement.

Assignment statement	Data type
`MyVar1 ← Total1 / Total2`	
`MyVar2 ← 27/10/2023`	
`MyVar3 ← "Sum1 / Sum2"`	
`MyVar4 ← Result1 AND Result2`	

[4]

(b) Other variables in the program have example values as shown:

Variable	Value
`Active`	`TRUE`
`Fraction`	`0.2`
`Code`	`"Ab12345"`

Complete the table by evaluating each expression using the example values.

Expression	Evaluates to
`Fraction >= 0.2 AND NOT Active`	
`INT((Fraction * 100) + 13.3)`	
`STR_TO_NUM(MID(Code, 4, 2)) + 5`	
`LENGTH("TRUE" & Code)`	

[4]

(c) The program makes use of complex statistical functions. The required functions are not built-in to the programming language and are too complicated for the programmer to write.

One solution would be to employ another programmer who has experience of writing these functions, as there is no time to train the existing programmer.

State **one other** way that these functions may be provided for inclusion in the program.

.....

..... [1]

(d) The hardware that runs the program is changed and the program needs to be modified so that it works with the new hardware.

Identify the type of maintenance that this represents **and** give **one other** reason why this type of maintenance may be needed.

Type .....

Reason .....

.....
[2]

2 `Data` is a 1D array of integers, containing 30 elements. All element values are unique.

(a) An algorithm will output the index of the element with the smallest value.

Draw a program flowchart to represent the algorithm.

[5]

(b) The 30 data values could have been stored in separate variables rather than in an array.

Explain the benefits of using an array when designing a solution to part **(a)**.

.....

..... [2]

(c) The requirement changes. Array `Data` needs to hold 120 elements and each value may include a decimal place.

Write a pseudocode statement to declare the modified array.

.....

..... [2]

3 The diagram represents a queue Abstract Data Type (ADT).

The organisation of this queue may be summarised as follows:

- The `FrontOfQueue` pointer points to the next data item to be removed.
- The `EndOfQueue` pointer points to the last data item added.

Queue

D3	← `FrontOfQueue`
D4	
D1	
D2	
D5	← `EndOfQueue`

The queue is implemented using three variables and a 1D array of eight elements as shown. The variable `NumItems` stores the number of items in the queue.

The pointer variables store indices (index numbers) of the array.

(a) Complete the diagram to represent the state of the queue as shown above.

Index	Array
1	
2	
3	
4	
5	
6	
7	
8	

Variable	
`FrontOfQueue`	
`EndOfQueue`	
`NumItems`	5

[3]

(b) A module `AddTo()` will add a value to the queue by manipulating the array and variables in part **(a)**.

The queue implementation is circular. When pointers reach the end of the queue, they will 'wrap around' to the beginning.

Before a value can be added to the queue, it is necessary to check the queue is not full.

The algorithm to add a value to the queue is expressed in six steps.

Complete the steps.

1. If `NumItems` ..... then jump to step 6.
2. Increment ..... .
3. If ..... then set `EndOfQueue` to ..... .
4. Increment ..... .
5. Set the ..... at the index stored in ..... to the ..... being added.
6. Stop.

[6]

4 A procedure `RandList()` will output a sequence of 25 random integers, where each integer is larger than the previous one.

(a) Write pseudocode for procedure `RandList()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) Procedure `RandList()` is modified so that the random numbers are also written into a 1D array `Result`.

A new module is written to confirm that the numbers in the array are in ascending order.

This module contains an `IF` statement that performs a comparison between elements:

```
IF (Result[x + 1] = Result[x]) OR (Result[x] > Result[x + 1]) THEN
   Sequence ← FALSE
ENDIF
```

Write a simplified version of the conditional clause.

.....

.....

..... [1]

5 A global 1D array of integers contains four elements, which are assigned values as shown:

```
Mix[1] ← 4
Mix[2] ← 2
Mix[3] ← 3
Mix[4] ← 5
```

A procedure `Process()` manipulates the values in the array.

The procedure is written in pseudocode as follows:

```
PROCEDURE Process(Start : INTEGER)
   DECLARE Value, Index, Total : INTEGER

   Index ← Start
   Total ← 0

   WHILE Total < 20
      Value ← Mix[Index]
      Total ← Total + Value

      IF Index < 4 THEN
         Mix[Index] ← Mix[Index] + Mix[Index+1]
      ELSE
         Mix[Index] ← Mix[Index] + Mix[1]
      ENDIF
      Index ← (Value MOD 4) + 1

   ENDWHILE

   Mix[1] ← Total * Index

ENDPROCEDURE
```

Complete the trace table on the opposite page by dry running the procedure when it is called as follows:

```
CALL Process(2)
```

Index	Value	Total	Mix[1]	Mix[2]	Mix[3]	Mix[4]

[6]

6 A function `TestNum()` will take a six-digit string as a parameter.

The function will test whether the string meets certain conditions and will return an integer value as follows:

Return value	Condition	Example
1	The last three digits are the same but non-zero.	"253444"
2	The last three digits are zero.	"253000"
3	The first three and last three digits are the same.	"410410"

The function will return the highest possible value for the given string.

If the string does not meet any of the conditions, zero is returned.

Write pseudocode for function `TestNum()`.

Assume that the parameter is valid.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

7 A structure chart shows the modular structure of a program:

(a) Explain the meaning of the curved arrow symbol which begins and ends at Module-A().

.....

.....

.....

..... [2]

(b) The structure chart shows that Sub-9() is a function.

A Boolean value is returned by Sub-9() for processing by Module-A().

The original parameter RA is of type integer and RB is of type string.

A record type `MyType` will be defined with three fields to store the values passed between the two modules.

(i) Write pseudocode to define `MyType`.

.....

.....

.....

.....

.....

..... [3]

(ii) The design is modified and Sub-9() is changed to a procedure.

The procedure will be called with a single parameter of type `MyType`.

Write the pseudocode header for procedure `Sub-9()`.

.....

..... [2]

8 A class of students are developing a program to send data between computers. Many computers are connected together to form a wired network. Serial ports are used to connect one computer to another.

Each computer:
- is assigned a unique three-digit ID
- has three ports, each identified by an integer value
- is connected to between one and three other computers.

Messages are sent between computers as a string of characters organised into fields as shown:

```
<STX><DestinationID><SourceID><Data><ETX>
```

Field name	Description
`STX`	a single character marking the start of the message (ASCII value 02)
`DestinationID`	three numeric characters identifying the destination computer
`SourceID`	three numeric characters identifying the source computer
`Data`	a variable length string containing the data being sent (Minimum length is 1 character)
`ETX`	a single character marking the end of the message (ASCII value 03)

For example, the following message contains the data "Hello Jack" being sent from computer "202" to computer "454":

```
<STX>"454202Hello Jack"<ETX>
```

Each computer will run a copy of the same program. Each program will contain a global variable `MyID` of type string which contains the unique ID of the computer in which the program is running.

The first two program modules are defined as follows:

Module	Description
`GetData()` (already written)	• returns the data field from a message that has been received • If no message is available, the module waits until one has been received.
`ReceiveFile()`	• takes a file name as a parameter of type string • creates a text file with the given file name (no checking required) • writes the data field returned by `GetData()` to the file • repeats until the data field is "****", which is not written to the file • outputs a final message giving the total number of characters written to the file, for example: `132456 characters were written to newfile.txt`

(a) Write pseudocode for module `ReceiveFile()`.

Module `GetData()` has already been written and must be used.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) The use of the string "****" as explained in the module description for `ReceiveFile()` may cause a problem.

Explain the problem and suggest a solution.

Problem .....

.....

.....

.....

Solution .....

.....

.....

.....

[3]

(c) Two new modules are defined, which will allow two users to exchange messages.

Module	Description

`Transmit()` (already written)	• takes two parameters:
	  ○ a string representing a message
	  ○ an integer representing a port number
	• transmits the message using the given port

`Chat()`	• takes two parameters:
	  ○ a string representing a Destination ID
	  ○ an integer representing a port number
	• extracts data from a received message using `GetData()` and outputs it
	• forms a message using data input by the user and sends it using `Transmit()`
	• repeats until either the output string or the sent string is `"Bye"`

Reminders:

- Each program contains a global variable `MyID` of type string which contains the unique ID of the computer in which the program is running.
- Messages are sent between computers as a string of characters organised into fields as shown:

```
<STX><DestinationID><SourceID><Data><ETX>
```

Write pseudocode for module `Chat()`.

Modules `GetData()` and `Transmit()` must be used.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(d) Module `GetData()` returns the data field from a message that has been received. If no message is available, the module waits until one has been received.

Explain the limitation of this on module `Chat()` from part **(c)**.

Describe a modification to `GetData()` to address this limitation.

Limitation .....

.....

.....

.....

Modification .....

.....

.....

.....

[3]

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2021**

INSERT **2 hours**

5427399501–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (PQ) 222380
© UCLES 2021

Note: An error occurs if a function call is not properly formed, or if the parameters are incorrect.

STRING Functions

`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"`

`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`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the lower case equivalent of `ThisChar`
Alphabetic characters that are not upper case are unchanged.

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

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the upper case equivalent of `ThisChar`
Alphabetic characters that are not lower case are unchanged.

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

`TO_UPPER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to upper case.

Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`

`TO_LOWER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to lower case.

Example: `TO_LOWER("JIM 803")` returns `"jim 803"`

© UCLES 2021

```
NUM_TO_STR(x : <data type1>) RETURNS <data type2>
```
returns a string representation of a numeric value.

Note: `<data type1>` may be `REAL` or `INTEGER`
`<data type2>` may be `CHAR` or `STRING`

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

```
STR_TO_NUM(x : <data type1>) RETURNS <data type2>
```
returns a numeric representation of a string.

Note: `<data type1>` may be `CHAR` or `STRING`
`<data type2>` may be `REAL` or `INTEGER`

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

```
IS_NUM(ThisString : STRING) RETURNS BOOLEAN
```
returns the value `TRUE` if `ThisString` represents a valid numeric value.

Example 1: `IS_NUM("12.36")` returns `TRUE`
Example 2: `IS_NUM("-12.36")` returns `TRUE`
Example 3: `IS_NUM("12.3a")` returns `FALSE`

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

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

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

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

NUMERIC Functions

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

Example: `INT(27.5415)` returns `27`

```
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`

OTHER Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: The function will generate an error if the file is not already open in `READ` mode.

Note: An error occurs if an operator with a value of an incorrect type is used.

OPERATORS

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to "Summer Pudding"
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

6646749220

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2022**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (KN/SW) 303772/3
© UCLES 2022

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** A programmer is testing a program using an Integrated Development Environment (IDE). The programmer wants the program to stop when it reaches a specific instruction or program statement in order to check the value assigned to a variable.

Give the technical term for the position at which the program stops.

..... [1]

(b) The following table lists some activities from the program development life cycle.

Complete the table by writing the life cycle stage for each activity.

Activity	Life cycle stage
An identifier table is produced.	
Syntax errors can occur.	
The developer discusses the program requirements with the customer.	
A trace table is produced.	

[4]

(c) An identifier table includes the names of identifiers used.

State **two other** pieces of information that the identifier table should contain.

1 .....

2 .....

[2]

(d) The pseudocode statements in the following table may contain errors.

State the error in each case or write 'NO ERROR' if the statement contains no error.

You can assume that none of the variables referenced are of an incorrect type.

Statement	Error
`Status ← TRUE AND FALSE`	
`IF LENGTH("Password") < "10" THEN`	
`Code ← LCASE("Electrical")`	
`Result ← IS_NUM(-27.3)`	

[4]

2 An algorithm is described as follows:

1. Input an integer value.
2. Jump to step 6 if the value is less than zero.
3. Call the function `IsPrime()` using the integer value as a parameter.
4. Keep a count of the number of times function `IsPrime()` returns `TRUE`.
5. Repeat from step 1.
6. Output the value of the count with a suitable message.

Draw a program flowchart to represent the algorithm.

START

END

[4]

3 **(a)** The module headers for five modules in a program are defined in pseudocode as follows:

Pseudocode module header
`FUNCTION Mod_V(S2 : INTEGER) RETURNS BOOLEAN`
`PROCEDURE Mod_W(P4 : INTEGER)`
`PROCEDURE Mod_X(T4 : INTEGER, BYREF P3 : REAL)`
`PROCEDURE Mod_Y(W3 : REAL, Z8 : INTEGER)`
`FUNCTION Mod_Z(F3 : REAL) RETURNS INTEGER`

An additional module `Head()` repeatedly calls three of the modules in sequence.

A structure chart has been partially completed.

(i) Complete the structure chart to include the information given about the six modules.

Do **not** label the parameters and do **not** write the module names.

[3]

(ii) Complete the table using the information in **part 3(a)** by writing each module name to replace the labels **A** to **F**.

Label	Module name
A	
B	
C	
D	
E	
F	

[3]

(b) The structure chart represents part of a complex problem. The process of decomposition is used to break down the complex problem into sub-problems.

Describe **three** benefits of this approach.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

4 **(a)** A procedure `LastLines()` will:

- take the name of a text file as a parameter
- output the **last** three lines from that file, in the same order as they appear in the file.

Note:

- Use local variables `LineX`, `LineY` and `LineZ` to store the three lines from the file.
- You may assume the file exists and contains **at least** three lines.

Write pseudocode for the procedure `LastLines()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) The algorithm in **part (a)** is to be amended. The calling program will pass the number of lines to be output as well as the name of the text file.

The number of lines could be any value from 1 to 30.

It can be assumed that the file contains **at least** the number of lines passed.

Outline **three** changes that would be needed.

1 .....

.....

.....

2 .....

.....

.....

3 .....

.....

.....

[3]

5 Study the following pseudocode. Line numbers are for reference only.

```
10 PROCEDURE Encode()
11   DECLARE CountA, CountB, ThisNum : INTEGER
12   DECLARE ThisChar : CHAR
13   DECLARE Flag : BOOLEAN
14   CountA ← 0
15   CountB ← 10
16   Flag ← TRUE
17   INPUT ThisNum
18   WHILE ThisNum <> 0
19      ThisChar ← LEFT(NUM_TO_STR(ThisNum), 1)
20      IF Flag = TRUE THEN
21         CASE OF ThisChar
22            '1' : CountA ← CountA + 1
23            '2' : IF CountB < 10 THEN
24                     CountA ← CountA + 1
25                  ENDIF
26            '3' : CountB ← CountB - 1
27            '4' : CountB ← CountB - 1
28                  Flag ← FALSE
29            OTHERWISE : OUTPUT "Ignored"
30         ENDCASE
31      ELSE
32         IF CountA > 2 THEN
33            Flag ← NOT Flag
34            OUTPUT "Flip"
35         ELSE
36            CountA ← 4
37         ENDIF
38      ENDIF
39      INPUT ThisNum
40   ENDWHILE
41   OUTPUT CountA
42 ENDPROCEDURE
```

(a) Procedure `Encode()` contains a loop structure.

Identify the type of loop **and** state the condition that ends the loop.

Do **not** include pseudocode statements in your answer.

Type .....

Condition .....

.....

[2]

(b) Complete the trace table below by dry running the procedure `Encode()` when the following values are input:

`12, 24, 57, 43, 56, 22, 31, 32, 47, 99, 0`

The first row is already complete.

ThisNum	ThisChar	CountA	CountB	Flag	OUTPUT
		0	10	TRUE	

[6]

(c) Procedure `Encode()` is part of a modular program. Integration testing is to be carried out on the program.

Describe **integration testing**.

.....

.....

.....

..... [2]

6 A string represents a series of whole numbers, separated by commas.

For example:

"12,13,451,22"

Assume that:

- the comma character ',' is used as a separator
- the string contains only the characters '0' to '9' and the comma character ','.

A procedure `Parse` will:

- take the string as a parameter
- extract each number in turn
- calculate the total value and average value of all the numbers
- output the total and average values with a suitable message.

Write pseudocode for the procedure.

```
PROCEDURE Parse(InString : STRING)
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
ENDPROCEDURE
```

[7]

7 A programming language has string functions equivalent to those given in the **insert**.

The language includes a `LEFT()` and a `RIGHT()` function, but it does **not** have a `MID()` function.

(a) Write pseudocode for an algorithm to implement your own version of the `MID()` function which will operate in the same way as that shown in the **insert**.

Do **not** use the `MID()` function given in the **insert**, but you may use any of the other functions.

Assume that the values passed to the function will be correct.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) The values passed to your `MID()` function in **part (a)** need to be validated.

Assume that the values are of the correct data type.

State **two** checks that could be applied to the values passed to the function.

1 .....

.....

2 .....

.....
[2]

BLANK PAGE

8 A program allows a user to save passwords used to log in to websites. A stored password is then inserted automatically when the user logs in to the corresponding website.

A global 2D array `Secret` of type `STRING` stores the passwords together with the website domain name where they are used. `Secret` contains 1000 elements organised as 500 rows by 2 columns.

Unused elements contain the empty string (`""`). These may occur anywhere in the array.

An example of a part of the array is:

Array element	Value
`Secret[27, 1]`	"thiswebsite.com"
`Secret[27, 2]`	"••••••••••••"
`Secret[28, 1]`	"thatwebsite.com"
`Secret[28, 2]`	"••••••••••••"

Note:

- For security, the passwords are stored in an encrypted form, shown as "••••••••••••" in the example.
- The passwords cannot be used without being decrypted.
- You may assume that the encrypted form of a password will **NOT** be an empty string.

The programmer has started to define program modules as follows:

Module	Description

`Exists()`	• Takes two parameters:
	  ○ a string
	  ○ a character
	• Performs a case-sensitive search for the character in the string
	• Returns `TRUE` if the character occurs in the string, otherwise returns `FALSE`

`Encrypt()`	• Takes a password as a parameter of type string • Returns the encrypted form of the password as a string
`Decrypt()`	• Takes an encrypted password as a parameter of type string • Returns the decrypted form of the password as a string

Note: in a case-sensitive comparison, 'a' is not the same as 'A'.

(a) Write pseudocode for the module `Exists()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) A new module `SearchDuplicates()` will:

- search for the **first** password that occurs more than once in the array and output a message each time a duplicate is found.

  For example, if the same password was used for the three websites ThisWebsite.com, website27.net and websiteZ99.org, then the following messages will be output:

```
"Password for ThisWebsite.com also used for website27.net"
"Password for ThisWebsite.com also used for websiteZ99.org"
```

- end once all messages have been output.

The module will output a message if no duplicates are found.
For example:

```
"No duplicate passwords found"
```

Write efficient pseudocode for the module `SearchDuplicates()`. `Encrypt()` and `Decrypt()` functions have been written.

Note: It is necessary to decrypt each password before checking its value.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(c) A password has a fixed format, consisting of **three groups of four** alphanumeric characters, separated by the hyphen character '-'.

An example of a password is:

"FxAf-3haV-Tq49"

Each password must:

- be 14 characters long
- be organised as three groups of four alphanumeric characters. The groups are separated by hyphen characters
- not include any duplicated characters, except for the hyphen characters.

An algorithm is needed for a new function `GeneratePassword()`, which will generate and return a password in this format.

Assume that the following modules have already been written:

Module	Description
`Exists()`	• Takes two parameters: ○ a string ○ a character • Performs a case-sensitive search for the character in the string • Returns `TRUE` if the character occurs in the string, otherwise returns `FALSE`
`RandomChar()`	• Generates a single random character from within one of the following ranges: ○ 'a' to 'z' ○ 'A' to 'Z' ○ '0' to '9' • Returns the character

Note: in a case-sensitive comparison, 'a' is not the same as 'A'.

Describe the algorithm for `GeneratePassword()`.

Do **not** use pseudocode statements in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

3263226235

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 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/CGW) 208684/3
© UCLES 2021

1 **(a)** An algorithm contains the following steps:

1. prompt and input a product number
2. validate the product number
3. save the validated product number to `FILE_A.txt`

(i) Write the pseudocode statement for step 1.

.....

.....[2]

(ii) State the purpose of `FILE_A.txt`.

.....

.....[1]

(b) State **two** benefits of creating program flowcharts when documenting a program.

1 .....

.....

2 .....

.....
[2]

(c) Draw **one** line from each technical term to its appropriate description.

Technical term	Description
Corrective maintenance	Stores data of the same data type in memory
Array	Amends an algorithm following identification of errors
Adaptive maintenance	Amends an algorithm in response to specification changes
Structure chart	Shows parameters passed between program modules

[3]

(d) The following pseudocode procedure, `SetupVars()`, initialises the variables used to check the status of a product.

```
PROCEDURE SetupVars()
   Description ← "CONCRETE-SLAB"
   Destination ← "ELY"
   Mileage ← 200
   EndOfYear ← 1
   Limit ← 20000
   Overdue ← FALSE
ENDPROCEDURE
```

Complete the table by evaluating each pseudocode expression.

If the expression is invalid, write “ERROR” in the **Evaluates to** column.

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

Pseudocode expression	Evaluates to
`EndOfYear * Limit / Mileage`	
`LENGTH(Description) / NUM_TO_STRING(Limit)`	
`MOD(20, LENGTH(Destination))`	
`(EndOfYear < 2) AND (Limit = 20000) AND NOT(Overdue)`	
`MID(Description, 1, 5) & LEFT(Destination, 2)` `& NUM_TO_STRING(Limit / 1000)`	

[5]

2 A company owns 50 cars that are available for hire. The cars are numbered from 1 to 50. The size of a car can be small, medium, or large.

The following pseudocode procedure, `SetOut()`, takes as input the size of the car the customer has requested for hire and outputs details of the cars that are available for hire for that size of car.

Line numbers are given for reference only.

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

```
01 DECLARE Available : ARRAY[1:50] OF BOOLEAN
02 DECLARE CarSize : ARRAY[1:50] OF STRING
03
04 PROCEDURE SetOut(Size: STRING)
05   DECLARE Index : REAL
06
07   FOR Index ← 0 TO 50
08     IF Size = CarSize[Index] AND Available[Index] = "TRUE"
09       THEN
10         OUTPUT NUM_TO_STRING(Index) & ": " & Size & " Available"
11     ENDIF
12   ENDFOR
13
14 ENDPROCEDURE
```

(a) Complete the table by identifying **four** line numbers that contain errors **and** give the correct pseudocode statement.

Line number	Correct pseudocode statement

[4]

(b) Study the pseudocode algorithm on page 4.

Describe the algorithm using **structured English**.

Do **not** use pseudocode or program code in your answer.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) The function `PayUsingAccount()` is required to process a payment for hiring a car as follows:

1. The function takes two parameters with `REAL` data type:

   - `Balance` is the amount of money in the account
   - `CostOfHire` is the cost of hiring the car.

2. If `CostOfHire` exceeds `Balance`, the rental is not authorised and `FALSE` is returned, otherwise `TRUE` is returned.

Write **pseudocode** for the function `PayUsingAccount()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

3 The function `AddressChecker()` checks if an email address is in the format:

<user>@<host>•<domain>

The design requirements of the function are:

- The <user> contains only lower-case characters.
- The <domain> contains exactly three alphabetic characters.

`AddressChecker()` will use three functions as described in the following table.

Function	Description
`Split()`	• Takes two parameters: ◦ a `STRING` value containing the email address ◦ a `BOOLEAN` value representing the required return value • Returns two different values: ◦ a `STRING` containing <user> if the `BOOLEAN` parameter value is `TRUE` ◦ a `STRING` containing <domain> if the `BOOLEAN` parameter value is `FALSE`
`IsLowerCase()`	• Takes an element of the email address as a `STRING` parameter • Returns a `BOOLEAN` value of `TRUE` if the string contains only lower-case characters, otherwise returns `FALSE`
`IsThreeLetter()`	• Takes the domain as a `STRING` parameter • Returns a `BOOLEAN` value of `TRUE` if the string consists of exactly three alphabetic characters, otherwise returns `FALSE`

`AddressChecker()` will:

- prompt and input an email address
- check that the email address is valid
- return the string `"Invalid"` if the email address does not meet the design requirements, otherwise return the email address.

(a) Draw a program flowchart to represent the algorithm for the `AddressChecker()` function.

Variable declarations are not required in program flowcharts.

[6]

(b) The function `AddressChecker()` is changed to check a list of email addresses stored in a text file.

The function's header is changed as follows:

```
FUNCTION AddressChecker(Filename : STRING, StartLine : INTEGER,
                        LastLine : INTEGER) RETURNS BOOLEAN
```

Assume that:

- The function is called with a valid filename of the text file.
- The text file exists and contains at least one email address.
- Each email address is stored in a separate line in the text file.
- The function does not contain logical errors.

Give **two** reasons why the function may return an unexpected result.

Reason 1 .....

.....

Reason 2 .....

.....
[2]

(c) **(i)** Explain what is meant by the term **syntax error**.

.....

.....

.....

..... [2]

(ii) Give an example of a syntax error in **program code**.

Programming language .....

Program code

.....

.....

.....

..... [1]

(d) The following incomplete structure chart shows part of the design of the function `AddressChecker()`.

Complete the structure chart.

[3]

(e) State **and** describe **two** features of an Integrated Development Environment (IDE) that aid debugging.

Feature 1 .....

.....

Description .....

.....

.....

Feature 2 .....

.....

Description .....

.....

.....

[4]

4 The following pseudocode contains a function that searches a text file for the string contained in the parameter `Match`.

```
DECLARE MatchItems : ARRAY[1:100] OF STRING

FUNCTION Extract(FileName : STRING, Match : STRING,
                 Signal : INTEGER) RETURNS BOOLEAN

   DECLARE FileLine : INTEGER
   DECLARE Counter : INTEGER
   DECLARE FileData, SubMatch : STRING
   DECLARE Result : BOOLEAN

   Counter ← 0
   FileLine ← 1
   Result ← FALSE

   OPENFILE FileName FOR READ

   WHILE NOT EOF(FileName) AND FileLine <= 100
      READFILE FileName, FileData
      SubMatch ← LEFT(FileData, LENGTH(Match))
      IF SubMatch = Match
         THEN
            MatchItems[Counter] ← FileData
            Counter ← Counter + 1
       ENDIF
       FileLine ← FileLine + 1
   ENDWHILE

   CLOSEFILE FileName
   IF Counter >= Signal
      THEN
        Result ← TRUE
   ENDIF
   RETURN Result

ENDFUNCTION
```

(a) Complete the trace table when the function is called using this statement:

```
Present ← Extract("DATA.txt", "TG12367", 1)
```

You may assume that the text file `DATA.txt` contains the following lines:

```
"XD43668#23/11/19#DSCP"
"TG12367#24/01/19#MAHA"
"HD44356#24/11/19#MAHA"
"TG12367#24/11/19#GHFI"
```

FileLine	Counter	LEFT(FileData, LENGTH(Match))	SubMatch = Match	Result

[5]

(b) The function is changed as follows:

- The statement on line 25 is changed to compare the last ten characters of `FileData` and the parameter `Match`.

  If the comparison is successful:

  - store the first seven characters of `FileData` in a global 1D array called `Original`
  - store the last four characters of `FileData` in a global 1D array called `Backup`

- The count of the number of items stored in the array `Backup` is returned.

(i) Rewrite the function header in **pseudocode** so that it meets the new requirements.

.....

.....

.....

..... [2]

(ii) Rewrite the `WHILE` structure in **pseudocode** so that it meets the new requirements. The `WHILE` structure is given here for reference.

```
WHILE NOT EOF(FileName) AND FileLine <= 100
   READFILE FileName, FileData
   SubMatch ← LEFT(FileData, LENGTH(Match))
   IF SubMatch = Match
      THEN
         MatchItems[Counter] ← FileData
         Counter ← Counter + 1
    ENDIF
    FileLine ← FileLine + 1
ENDWHILE
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

Question 4 continues on the next page.

(c) A procedure, `Encrypt()`, is required to encrypt the data stored in each line of the text file `DATA.txt`.

You may assume that the text file `DATA.txt` contains the following lines:

```
"XD43668#23/11/19#DSCP"
"TG12367#24/01/19#MAHA"
"HD44356#24/11/19#MAHA"
"TG12367#24/11/19#GHFI"
```

The requirements of the procedure are:

- the day of the month from the line of data is extracted as the encryption key
- the encryption key is added to the ASCII value of each character of the line to form the encrypted data
- the encrypted data are appended to the text file `DATA-EN.txt` until there are no more lines in the text file `DATA.txt`

Write **program code** for the procedure `Encrypt()`.

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]

5 A geocode string contains four alphanumeric characters, followed by the character `'+'` and then two alphanumeric characters.

For example, the geocode string for the Avocado Restaurant in Cambridge UK is:

```
"1DFG+9N"
```

A website allows a user to upload a review of a restaurant.

A programmer has defined the following two arrays:

Array	Description
`GeoData`	• a global 1D array of data type `STRING` • stores the geocode for each restaurant Example: `GeoData[1] ← "1DFG+9N"`

`Review`	• a global 1D array of data type `STRING`
	• stores the following data items as a single element in the array:
	  ◦ the four-character user ID of the reviewer
	  ◦ the geocode of the restaurant
	  ◦ the text of the review
	• the hash symbol (#) separates the data items in each single element
	
	Example of a review for the Avocado restaurant submitted by user ID JH76:
	`Review[1000] ← "JH76#1DFG+9N#Good restaurant"`

Assume that:

- both arrays are initialised with an empty string
- both arrays can store up to 20 000 elements
- new reviews are added in the next unused element of the `Review` array.

(a) The programmer writes a function `CheckReview()`.

The design requirements of the function are shown in the following table:

Description	Parameter	Returns
• searches the `GeoData` array to determine if the user's latest review contains a listed geocode	• a `STRING` value containing the user ID of the reviewer	• the comment from the review if the geocode exists in the `GeoData` array • a `STRING` value of `"LOCATION NOT FOUND"` if the geocode of the user's review does not exist in the `GeoData` array • a `STRING` value of `"NO REVIEW"` if no review exists for the user ID

Write **pseudocode** to complete the function `CheckReview()`.

```
DECLARE GeoData : ARRAY[1:20000] OF STRING
DECLARE Review : ARRAY[1:20000] OF STRING
FUNCTION CheckReview(UserID : STRING) RETURNS STRING
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
ENDFUNCTION
```

[8]

(b) Before a review is published on the website, it is checked to find out whether the review contains a valid geocode.

The function `AddReview()` has the following requirements:

Description	Parameters	Returns
• calls `CheckReview()` • if the text of the review is returned from `CheckReview()`, store the text in the global 1D array `Accepted`	• a `STRING` value containing the user ID of the reviewer	• a `BOOLEAN` value of `FALSE` if the text of the review is not added in the array `Accepted`, otherwise returns a `BOOLEAN` value of `TRUE`

Assume that:

- the 1D array `Accepted` is initialised with empty strings
- the array contains up to 20 000 elements.

Write **program code** for the function `AddReview()`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) When reviewers write a review for a restaurant, they give the restaurant a score out of 10.

The 2D global array `ReviewScores` stores the review data as follows:

- The first column element stores the seven-character alphanumeric geocode string of the restaurant with its average score, separated by the hash symbol (#).
- The second column element stores the comment from the review that has the highest score.

The array has dimensions of 20 000 rows and 2 columns.

For example, the contents of the 2D array `ReviewScores` for review `20` are:

```
ReviewScores[20, 1] ← "1DFG+9N#8.3"
ReviewScores[20, 2] ← "Food was good but I waited too long"
```

The procedure `BestRestaurants()` is required as follows:

Description	Parameter	Output
• searches the array `ReviewScores` for restaurants with an average score that exceeds `8.0`	• a `STRING` value consisting of the geocode of the restaurant	• if the parameter matches a geocode in the array `ReviewScores` and the average score exceeds `8.0`, output the geocode concatenated with the review comment

Write **pseudocode** for the procedure `BestRestaurants()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

Appendix

Built-in functions (pseudocode)

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

```
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'`

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

```
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"`

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

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

```
MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

```
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: `STR_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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

3033251826

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2023**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (EF/SG) 312086/3
© UCLES 2023

Refer to the **insert** for the list of pseudocode functions and operators.

1 A program calculates the postal cost based on the weight of the item and its destination. Calculations occur at various points in the program and these result in the choice of several possible postal costs. The programmer has built these postal costs into the program.

For example, the postal cost of $3.75 is used in the following lines of pseudocode:

```
IF Weight < 250 AND ValidAddress = TRUE THEN
   ItemPostalCost ← 3.75    // set postal cost for item to $3.75
   ItemStatus ← "Valid"     // item can be sent
ENDIF
```

(a) **(i)** Identify a more appropriate way of representing the postal costs.

..... [1]

(ii) Describe the advantages of your answer to **part (a)(i)** with reference to this program.

.....

.....

.....

.....

.....

..... [3]

(b) The lines of pseudocode contain features that make them easier to understand.

State **three** of these features.

1 .....

2 .....

3 .....
[3]

(c) Give the **appropriate** data types for the following variables:

`ValidAddress` .....

`ItemPostalCost` .....

`ItemStatus` .....
[3]

2 A program stores a user's date of birth using a variable `MyDOB` of type `DATE`.

(a) Write a pseudocode statement, using a function from the **insert**, to assign the value corresponding to 17/11/2007 to `MyDOB`.

..... [1]

(b) `MyDOB` has been assigned a valid value representing the user's date of birth.

Write a pseudocode statement to calculate the number of months from the month of the user's birth until the end of the year and to assign this to the variable `NumMonths`.

For example, if `MyDOB` contains a value representing 02/07/2008, the value 5 would be assigned to `NumMonths`.

..... [2]

(c) The program will output the day of the week corresponding to `MyDOB`.

For example, given the date 22/06/2023, the program will output `"Thursday"`.

An algorithm is required. An array will be used to store the names of the days of the week.

Define the array **and** describe the algorithm in **four** steps.

Do **not** use pseudocode statements in your answer.

Array definition .....

.....

Step 1 .....

.....

.....

Step 2 .....

.....

.....

Step 3 .....

.....

.....

Step 4 .....

.....

.....

[6]

3 A program stores data in a text file. When data is read from the file, it is placed in a queue.

(a) The diagram below represents an Abstract Data Type (ADT) implementation of the queue. Each data item is stored in a separate location in the data structure. During initial design, the queue is limited to holding a maximum of 10 data items.

The operation of this queue may be summarised as follows:

- The Front of Queue Pointer points to the next data item to be removed.
- The End of Queue Pointer points to the last data item added.
- The queue is circular so that locations can be reused.

0		
1		
2		
3		
4		
5	Red	← Front of Queue Pointer
6	Green	
7	Blue	
8	Pink	← End of Queue Pointer
9		

(i) Describe how the data items Orange and Yellow are added to the queue shown in the diagram.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(ii) The following diagram shows the state of the queue after several operations have been performed. All queue locations have been used at least once.

0	D4	
1	D3	← End of Queue Pointer
2	D27	
3	D8	
4	D33	
5	D17	← Front of Queue Pointer
6	D2	
7	D1	
8	D45	
9	D60	

State the number of data items in the queue.

..... [1]

(b) The design of the queue is completed and the number of locations is increased.

A function `AddToQueue()` has been written. It takes a string as a parameter and adds this to the queue. The function will return `TRUE` if the string was added successfully.

A procedure `FileToQueue()` will add each line from the file to the queue. This procedure will end when all lines have been added or when the queue is full.

Describe the algorithm for procedure `FileToQueue()`.

Do **not** use pseudocode in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

4 A function `GetNum()` will:

1. take two parameters: a string and a character
2. count the number of times that the character occurs in the string
3. return the count.

Any comparison between characters needs to be case sensitive. For example, character '`a`' and character '`A`' are not identical.

Write pseudocode for function `GetNum()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

BLANK PAGE

5 A programmer has produced the following pseudocode to output the square root of the numbers from 1 to 10.

Line numbers are for reference only.

```
10  DECLARE Num : REAL
11  Num ← 1.0
...
40  REPEAT
41     CALL DisplaySqrt(Num)
42     Num ← Num + 1.0
43  UNTIL Num > 10
...
50  PROCEDURE DisplaySqrt(BYREF ThisNum : REAL)
51     OUTPUT ThisNum
52     ThisNum ← SQRT(ThisNum)       // SQRT returns the square root
53     OUTPUT " has a square root of ", ThisNum
54  ENDPROCEDURE
```

The pseudocode is correctly converted into program code.

Function `SQRT()` is a library function and contains no errors.

The program code compiles without errors, but the program gives unexpected results. These are caused by a design error.

(a) Explain why the program gives unexpected results.

.....

.....

.....

.....

.....

..... [3]

(b) Explain why the compiler does **not** identify this error.

.....

..... [1]

(c) Describe how a typical Integrated Development Environment (IDE) could be used to identify this error.

.....

.....

.....

.....

.....

..... [3]

(d) The pseudocode is converted into program code as part of a larger program.

During compilation, a complex statement generates an error.

The programmer does not want to delete the complex statement but wants to change the statement so that it is ignored by the compiler.

State how this may be achieved.

.....

..... [1]

6 A procedure `Square()` will take an integer value in the range 1 to 9 as a parameter and output a number square.

The boundary of a number square is made up of the character representing the parameter value. The inside of the number square is made up of the asterisk character (`*`).

Parameter value	**1**	**2**	**3**	**4**	**...**	**9**

Output	`1`	`22`
		`22`
			`333`
			`3*3`
			`333`
				`4444`
				`4**4`
				`4**4`
				`4444`
					...	`999999999`
						`9*******9`
						`9*******9`
						`9*******9`
						`9*******9`
						`9*******9`
						`9*******9`
						`9*******9`
						`999999999`

The pseudocode `OUTPUT` command starts each output on a new line. For example, the following three `OUTPUT` statements would result in the outputs as shown:

```
OUTPUT "Hello"
OUTPUT "ginger"
OUTPUT "cat"
```

Resulting output:

```
Hello
ginger
cat
```

Write pseudocode for procedure `Square()`.

Parameter validation is not required.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

BLANK PAGE

7 A computer system for a shop stores information about each customer. The items of information include name and address (both postal and email) together with payment details and order history. The system also stores the product categories they are interested in and how they would like to be contacted.

(a) The shop wants to add a program module that will generate emails to be sent to customers who may be interested in receiving details of new products.

(i) State **three** items of information that the new module would need. Justify your choice in each case.

Information .....

Justification .....

.....

Information .....

Justification .....

.....

Information .....

Justification .....

.....
[3]

(ii) Identify **two** items of customer information that would **not** be required by the new module. Justify your choice in each case.

Information .....

Justification .....

.....

Information .....

Justification .....

.....
[2]

(b) The program includes a module to validate a Personal Identification Number (PIN). This is used when customers pay for goods using a bank card.

A state-transition diagram has been produced for this module.

The table show the inputs, outputs and states for this part of the program:

Current state	Input	Output	Next state
S1	Input PIN		S2
S2	Re-input PIN	Display error	S2
S2	Cancel	Re-prompt	S1
S2	Valid PIN	Enable payment	S4
S2	Too many tries	Block Account	S3

Complete the state-transition diagram to represent the information given in the table.

[4]

8 A computer shop assembles computers using items bought from several suppliers. A text file `Stock.txt` contains information about each item.

Information for each item is stored as a single line in the `Stock.txt` file in the format:

```
<ItemNum><SupplierCode><Description>
```

Valid item information is as follows:

	Format	Comment
`ItemNum`	4 numeric characters	unique number for each item in the range "0001" to "5999" inclusive
`SupplierCode`	3 alphabetic characters	to identify the supplier of the item
`Description`	a string	a minimum of 12 characters

The file is organised in ascending order of `ItemNum` and does **not** contain all possible values in the range.

A programmer has started to define program modules as follows:

Module	Description
`OnlyAlpha()` (already written)	• called with a parameter of type string • returns `TRUE` if the string contains only alphabetic characters, otherwise returns `FALSE`
`CheckInfo()`	• called with a parameter of type string representing a line of item information • checks to see whether the item information in the string is valid • returns `TRUE` if the item information is valid, otherwise returns `FALSE`

(a) Write pseudocode for module `CheckInfo()`.

Module `OnlyAlpha()` should be used as part of your solution.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) A new module is defined as follows:

Module	Description
`AddItem()`	• called with a parameter of type string representing valid information for a new item that is not currently in the `Stock.txt` file • creates a new file `NewStock.txt` from the contents of the file `Stock.txt` and adds the new item information at the appropriate place in the `NewStock.txt` file

As a reminder, the file `Stock.txt` is organised in ascending order of `ItemNum` and does not contain all possible values in the range.

Write pseudocode for module `AddItem()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(c) The program contains modules `SuppExists()` and `CheckSupplier()`. These have been written but contain errors. These modules are called from several places in the main program and testing of the main program (integration testing) has had to stop.

Identify a method that can be used to continue testing the main program before the errors in these modules have been corrected **and** describe how this would work.

Method .....

Description .....

.....

.....

.....
[3]

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2018**

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 **16** printed pages.

DC (NF/JG) 146141/3
© UCLES 2018

CAMBRIDGE International Examinations

1 **(a)** A farm has a number of greenhouses used to grow vegetables. Each greenhouse has a different identification number. A program is needed to store temperature information for each greenhouse throughout the day.

Give a suitable **identifier name** for each of the data items.

Description of data item	Suitable identifier name
The temperature inside the greenhouse	
The temperature outside the greenhouse	
The greenhouse identification number	
The time of the temperature measurements	

[4]

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

Variable	Value
`Mark`	`60`
`Subject`	`"Computer Science"`
`Grade`	`'B'`
`CourseCompleted`	`TRUE`
`AverageMark`	`49.5`

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 16.

Expression	Evaluates to
`"Fas" & MID(Subject, 6, 3)`	
`LEFT(Mark, 1)`	
`10 + ASC(Grade)`	
`MOD(AverageMark * 2, 3)`	
`CourseCompleted AND (Mark >= 60)`	

[5]

(ii) Programming languages support different data types.

Give an appropriate data type for each of these variables from **part (b)(i)**.

Variable	Data type
`Mark`	
`Subject`	
`Grade`	
`CourseCompleted`	
`AverageMark`	

[5]

2 The following is a function design in pseudocode.

Line numbers are given for reference only.

```
01  FUNCTION CountDigits(InString : STRING) RETURNS CHAR
02
03  DECLARE nc : CHAR
04  DECLARE c : INTEGER
05  DECLARE n : INTEGER
06
07
08  c ← 0
09  n ← LENGTH(InString)   // get number of characters for loop
10
11  WHILE n > 0            // repeat until no more characters left
12
13  nc ← LEFT(InString,1)
14  n ← n - 1
15  InString ← RIGHT(InString,n)  // remove first character
16
17  IF (nc < '0') OR (nc > '9')
18  THEN
19  // do nothing
20  ELSE
21  c ← c + 1
22  ENDIF
23
24  ENDWHILE
25
26  RETURN c
27
28  ENDFUNCTION
```

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

State **two** such features.

Feature 1 .....

Feature 2 .....
[2]

(ii) State **two** additional features that should be used to make this pseudocode easier to read and understand.

Feature 1 .....

Feature 2 .....
[2]

(b) Study the function `CountDigits()`. Identify the features of the function in the following table.

Feature	Answer
A line number containing an example of an assignment statement	
A line number containing the start of a 'pre-condition' loop	
A line number containing the end of a 'pre-condition' loop	
A line number containing the start of a selection statement	
The number of parameters passed to the `LEFT()` function	
The Boolean operator used	
The number of times the function `LEFT()` is called from within `CountDigits()` resulting from the call: `Result ← CountDigits("AB27C4")`	
The number of local variables	

[8]

(c) **(i)** There is a mistake in the pseudocode that would produce a data type mismatch error if a programmer were to write similar program code.

Describe this mistake and how it may be corrected.

.....

.....

.....

..... [2]

(ii) Lines 17 to 22 of the pseudocode could be written in a simplified form.

Re-write the lines in a simplified form.

.....

.....

.....

.....

.....

.....

.....

..... [4]

3 A chocolate factory produces bars of chocolate. A computer program controls the process.

The weight of each bar is stored in an array, `BarWeight`. The array contains 100 elements, representing the weights of 100 bars that make up one shipping box.

A procedure, `CheckWeight()`, is required to:

1. examine each array element and count how many times the weight has exceeded `MaxWeight`
2. compare the count obtained with a limit value, `Threshold`. Call procedure `ServiceCheck()` if the count exceeds the `Threshold`
3. output a message if the count does not exceed the `Threshold`. For example:

```
"ShippingBox OK - maximum weight exceeded 3 times."
```

Draw a program flowchart on the next page to represent the algorithm for the `CheckWeight()` procedure.

Assume that:

- the array contains 100 valid weight values and the first element is `BarWeight[1]`
- `MaxWeight`, `Threshold` and `BarWeight` are global variables.

Variable declarations are not required in program flowcharts.

[10]

Question 4 begins on the next page.

4 The structure chart shows part of the design of a program for an online shopping system.

(a) **(i)** Draw on the chart to show the following facts.

- Each of the modules at the lower level returns a Boolean parameter, X.
- Process basket will call only one of the modules shown at the lower level.

[2]

(ii) The parameters A to G shown on the chart will be used to pass the following information.

```
PaymentDetails
Quantity
BasketID
DeliveryAddress
ItemID
```

Complete the following table to show the parameter and the information it represents.

Parameter	Information
A	
B	
C	
D	
E	
F	
G	

[3]

5 A golf club holds information about its members. When a member completes a round of golf, their score is stored along with their membership number and the date of the round.

A program is to be written to store and process the score information. The information to be stored is formed into a string as follows:

```
<MembershipNumber><Date><Score>
```

(a) The program designer considers storing the strings in either a 1D array, `RoundScore` or as a separate variable for each round, for example, `RoundScore01`, `RoundScore02`, `RoundScore03` and so on.

Describe **two** advantages of storing the strings in a 1D array rather than as separate variables.

Advantage 1 .....

.....

Advantage 2 .....

.....
[2]

(b) A procedure, `AddNewScores()` is being developed. The procedure will be coded using an Integrated Development Environment (IDE).

Name **two** features provided by an IDE that assist in the **initial detection of errors** in the procedure.

Feature 1 .....

.....

Feature 2 .....

.....
[2]

(c) The program needs a function, `GetNumber()`, to return a valid membership number. A valid membership number is a four-digit numeric string between “1111” and “9999”.

The structured English representing the algorithm for this function is as follows:

1. Prompt and input a membership number.
2. Validate the membership number.
3. Repeat from step 1 if the membership number is invalid.
4. Return the valid membership number as a string.

An example of the function call in pseudocode is:

```
MembershipNumber ← GetNumber()
```

Write **program code** for the `GetNumber()` function.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(d) The program designer decides to store the strings from **part (a)** in a text file, `ScoreDetails.txt`

Each string is formatted as follows:

```
<MembershipNumber><Date><Score>
```

- `MembershipNumber` is a four-digit numeric string between `"1111"` and `"9999"`.
- `Date` is a six-digit numeric string in the format DDMMYY, for example `"040716"`.
- `Score` is a two-digit numeric string in the range `"50"` to `"99"`.

A procedure `OutputLowestScore()` is required to output the lowest score for an individual member.

Assume that there is at least one string stored for each member.

The program needs a procedure, `OutputLowestScore()`, to process the `ScoreDetails.txt` file and output the lowest score for an individual member.

1. Use the `GetNumber()` function to obtain a valid membership number.
2. Search the `ScoreDetails.txt` file for the lowest score for that member.
3. Output a message giving the lowest score for that member and the date of the round. For example: `"The lowest score was 66 on 300917"`

Write **program code** for the `OutputLowestScore()` procedure.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [10]

6 **(a)** Individual elements in a 1D array are referenced using an integer value that is used as the subscript to the array.

Give the technical terms for the minimum and maximum values the subscript may take.

Minimum value .....

Maximum value .....
[2]

(b) A 2D array, `Picture`, contains data representing a bitmap image. Each element of the array represents one pixel of the image. The image is grey-scale encoded where the value of each pixel ranges from 0 (representing black) to 255 (representing white) with intermediate values representing different levels of grey.

A graphics program needs a procedure, `Flip()`, to flip (reflect) the image. An example of an image before and after the function is:

Before flip

After flip

The values contained in the 2D array before the flip are as follows:

Data values

80	80	255	80	80	255	80	80
80	80	255	80	80	255	80	80
255	80	120	120	120	120	255	80
255	80	255	255	255	255	80	80
255	80	120	120	120	120	80	80

In pseudocode, the array is declared as follows:

```
DECLARE Picture : ARRAY[1:5, 1:8] OF INTEGER
```

Write **pseudocode** to implement the `Flip()` procedure.

Assume that `Picture` is a global variable.

..... [8]

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 string `"BCD"`

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

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

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

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

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

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

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`

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2018

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2021**

INSERT **2 hours**

7483199901–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (PQ) 200597/1
© UCLES 2021

Note: An error occurs if a function call is not properly formed, or if the parameters are incorrect.

STRING Functions

`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"`

`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`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the lower case equivalent of `ThisChar`
Characters that are not upper case alphabetic are returned unchanged

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

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the upper case equivalent of `ThisChar`
Characters that are not lower case alphabetic are returned unchanged

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

`TO_UPPER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to upper case

Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`

`TO_LOWER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to lower case

Example: `TO_LOWER("JIM 803")` returns `"jim 803"`

`NUM_TO_STR(x : <data type>) RETURNS STRING`
returns a string representation of a numeric value
Note: `<data type>` may be `REAL` or `INTEGER`

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

```
STR_TO_NUM(x : <data type1>) RETURNS <data type2>
```
returns a numeric representation of a string
Note: `<data type1>` may be `CHAR` or `STRING`
Note: `<data type2>` may be `REAL` or `INTEGER`

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

```
IS_NUM(ThisString : STRING) RETURNS BOOLEAN
```
returns the value `TRUE` if `ThisString` represents a valid numeric value
Note: `<data type>` may be `CHAR` or `STRING`

Example: `IS_NUM("12.36")` returns `TRUE`
Example: `IS_NUM("-12.36")` returns `TRUE`
Example: `IS_NUM("12.3a")` returns `FALSE`

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

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

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

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

NUMERIC Functions

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

Example: `INT(27.5415)` returns `27`

```
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`

DATE Functions

Note: Date format is assumed to be DDMMYYYY unless otherwise stated.

```
DAY(ThisDate : DATE) RETURNS INTEGER
```
returns the current day number from `ThisDate`

Example: `DAY(4/10/2003)` returns `4`

```
MONTH(ThisDate : DATE) RETURNS INTEGER
```
returns the current month number from `ThisDate`

Example: `MONTH(4/10/2003)` returns `10`

`YEAR(ThisDate : DATE) RETURNS INTEGER`
returns the current year number from `ThisDate`

Example: `YEAR(4/10/2003)` returns `2003`

`DAYINDEX(ThisDate : DATE) RETURNS INTEGER`
returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2, Tuesday = 3 etc.

Example: `DAYINDEX(12/05/2020)` returns `3`

`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE`
returns a variable of type `DATE`

`NOW() RETURNS DATE`
returns the current date

OTHER Functions

`EOF(FileName : STRING) RETURNS BOOLEAN`
returns `TRUE` if there are no more lines to be read from file `FileName`

Note: This function will generate an ERROR if the file is not already open in `READ` mode

OPERATORS

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: This operator may also be used to concatenate a character with a string
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example `10 DIV 3` evaluates to `3`

Note: An error is generated if an operator is used with a value or values of an incorrect type.

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.

© UCLES 2021

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

2054275341

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2016**

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 **20** printed pages.

DC (LK/FD) 127385
© UCLES 2016

CAMBRIDGE International Examinations

There is an **Appendix** on pages 19 and 20. Some questions will refer you to this information.

1 A programmer wants to write a program to calculate the baggage charge for a passenger's airline flight.

Two types of ticket are available for a flight:

- economy class (coded E)
- standard class (coded S)

Each ticket type has a baggage weight allowance as shown below. The airline makes a charge if the weight exceeds the allowance.

Ticket type	Baggage allowance (kg)	Charge rate per additional kg ($)
'E'	16	3.50
'S'	20	5.75

(a) A program flowchart will document the program. The flowchart will contain the following statements:

Statement number	Statement
1	`Charge ← 0`
2	`INPUT BaggageWeight`
3	`Charge ← ExcessWeight * ChargeRate`
4	`Is ExcessWeight > 0 ?`
5	`INPUT TicketType`
6	`ExcessWeight ← BaggageWeight - BaggageAllowance`
7	`BaggageAllowance ← 16`
8	`ChargeRate ← 3.5`
9	`OUTPUT Charge`
10	`ChargeRate ← 5.75`
11	`BaggageAllowance ← 20`
12	`Is TicketType = 'E' ?`

Complete the flowchart by putting the appropriate **statement number** in each flowchart symbol. Statement 5 has been done for you.

[6]

(b) The programmer needs data to test the flowchart.

Complete the table of test data below to show **five** tests.

`TicketType`	`BaggageWeight`	Explanation	Expected output
`E`	`15`	....	
		....	
		....	
		....	
		....	

[5]

(c) The program design is to be amended. The value input by the user for the ticket type is to be validated. Part of the amended flowchart is shown below.

Write **pseudocode** to use a pre-condition loop for this validation.

.....

.....

.....

.....

.....

.....[3]

2 A sensing device sends bit values to a computer along data channels.

- Channel 1 transmits a sequence of binary values from a sensor
- Channel 2 transmits at regular intervals to indicate whether the sensor is switched on or off:
  - 0 indicates switched off
  - 1 indicates switched on

A program tests the bits received from the sensing device.

A program reads the signal from Channel 2 after every six values from Channel 1.

A built-in function `READ(<ChannelNumber>)` reads a value from the specified channel.

Pseudocode for the program is as follows:

```
BitCount ← 0
Status2 ← READ(2)
WHILE Status2 = 1

  FOR ReadingCount ← 1 TO 6
    ThisBit ← READ(1)
    IF ThisBit = 1
      THEN
        BitCount ← BitCount + 1
    ENDIF
    IF BitCount = 5
      THEN
        OUTPUT "Error - Investigate"
        BitCount ← 0
    ENDIF
  ENDFOR

  Status2 ← READ(2)
ENDWHILE
```

(a) Trace the execution of the program for the following sequence of bits.

Channel 1		1	0	1	1	1	0		1	1	0	0	1	1	
Channel 2	1							1							0

Status2	ReadingCount	ThisBit	BitCount	OUTPUT
			0	
1	1	1	1	
	2			

[7]

(b) Identify the following constructs in the given program, using line numbers.

For multi-line constructs give the first line number only.

Construct	Line number
Assignment	
Selection	
Iteration	

[3]

3 You will need to refer to the list of pseudocode string-handling functions in the **Appendix**.

ASCII code table (part)					
Character	Decimal	Character	Decimal	Character	Decimal
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

(a) For each statement, write the value assigned to the variable.

(i) `Term ← CHARACTERCOUNT("TSUNAMI")`

`Term` .....[1]

(ii) `Answer1 ← ASC('G') + ASC(<Space>)`

`Answer1` .....[1]

(iii) `Answer2 ← CHR(CHARACTERCOUNT("HELLO") + 70)`

`Answer2` .....[1]

(iv) `Word ← SUBSTR("Welcome home", 4, 7) )`

`Word` .....[1]

Question 3(b) continues on page 10.

(b) A programmer wants to design a procedure to calculate a customer ID number from the customer's surname.

The procedure will:

- input the surname
- isolate each character in the surname and find the corresponding ASCII code
- calculate the total of all these ASCII codes
- this total is the customer ID

(i) Complete the pseudocode for this procedure.

You will need to refer to the list of pseudocode string-handling functions in the Appendix.

```
PROCEDURE CalculateCustomerID

  OUTPUT "Key in surname"

  INPUT Surname

  Length ← ..................................................................................

  CustomerID ← 0

  FOR i ← 1 TO Length

    // NextChar is a single character from Surname

    NextChar ← ..............................................................................

    NextCodeNumber ← ASC(NextChar)

    CustomerID ← CustomerID + ...............................................................

  ENDFOR

  OUTPUT "Customer ID is ", CustomerID
```

[3]

(ii) Write **program code** for procedure `CalculateCustomerID`.

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

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

(c) The programmer decides that it would be better to write the procedure as a function. The user will now input the surname in the main program.

Write **program code** for the following:

State your programming language .....

(i) The function header for this new function `CalculateCustomerID`

.....[3]

(ii) The additional statement required within the function body to complete the change from a procedure to a function.

.....

.....[1]

(iii) The statement in the main program which:

- calls the function for surname Wilkes
- assigns the result to variable `ThisID`

.....[3]

(d) **(i)** The new function `CalculateUserID` is an example of a 'user-defined function'.

State **two** differences between a built-in function and a user-defined function.

1 .....

.....

2 .....

.....[2]

(ii) State **two** things that built-in and user-defined functions have in common.

1 .....

.....

2 .....

.....[2]

4 A company employs Ahmed as a programmer.

(a) At College, before joining the company, Ahmed used two items of software for programming:

- a text editor
- a compiler

Describe how he could have developed programs using these software tools.

Include in the description the terms 'object code' and 'source code'.

.....

.....

.....

.....

.....

.....

.....[3]

(b) Ahmed now uses an Integrated Development Environment (IDE) for programming.

(i) State **one** feature an IDE provides to help with the identification of syntax errors.

.....

.....[1]

(ii) State **one** feature an IDE provides to carry out white box testing.

.....

.....[1]

(c) The company maintains a file of product data. Ahmed is to write a program to add a new product and search for a product based on the structure diagram shown:

The program records the following data for each product:

- product code
- product description
- product retail price

The text file `PRODUCTS` stores each data item on a separate line, as shown below:

File **PRODUCTS**

0198
Plums(10kg)
11.50
0202
Onions(20kg)
10.00
~
0376
Mango chutney(1kg)
02.99
~
0014
Mango(10kg)
12.75

The program uses the variables shown in the identifier table.

Identifier	Data type	Description
PRODUCTS	TEXT FILE	Storing the code, description and retail price for all current products
PCode	ARRAY[1:1000] OF STRING	Array storing the product codes
PDescription	ARRAY[1:1000] OF STRING	Array storing the product descriptions
PRetailPrice	ARRAY[1:1000] OF REAL	Array storing the product retail prices
i	INTEGER	Array index used by all three arrays

(i) The first operation of the program is to read all the product data held in file `PRODUCTS` and write them into the three 1D arrays.

Complete the pseudocode below.

```
OPEN ..............................................................................

i ← 1

WHILE ..............................................................................

  READFILE ("PRODUCTS", ..............................................................)

  READFILE ("PRODUCTS", ..............................................................)

  READFILE ("PRODUCTS", ..............................................................)

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

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

ENDWHILE

CLOSE "PRODUCTS"

OUTPUT "Product file contents written to arrays"
```

[5]

When Ahmed designed the `PRODUCTS` file, he considered the alternative file structure shown opposite.

It stores one product per line in the text file.

File **PRODUCTS**

0198 Plums(10kg) 11.50
0202 Onions(20kg) 10.00
⋮
0376 Mango chutney(1kg) 02.99
⋮
0014 Mango(10kg) 12.75

(ii) State **one** benefit and **one** drawback of this file design.

Benefit .....

.....

Drawback .....

.....[2]

(d) To code the 'Search by product code' procedure, Ahmed draws a structure chart showing the different stages.

The procedure uses the variables shown in the identifier table.

Identifier	Data type	Description
`SearchCode`	`STRING`	Product code input by the user
`ThisIndex`	`INTEGER`	Array index position for the corresponding product
`ThisDescription`	`STRING`	Product description found
`ThisRetailPrice`	`REAL`	Product retail price found

You can assume that before the procedure is run, all the product data is read from file `PRODUCTS` and then stored in three 1D arrays as described in **part (c)(i)**.

Label the structure chart to show the input(s) and output(s).

[4]

(e) A first attempt was made at writing the 'Search for product code' module.
Ahmed designs this as a function `ProductCodeSearch`.

The function returns an integer value as follows:

- if the product code is found, it returns the index position of the 1D array `PCode` being searched
- if the product code is not found, the function returns -1

Write **program code** for function `ProductCodeSearch`.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

5 Study the following pseudocode statements.

```
CONST  Pi =  3.1          : REAL

DECLARE Triangle, Base, Height, Radius, Cone : REAL

DECLARE a, b, c, Answer2 : INTEGER

DECLARE Answer1           : BOOLEAN

Base ← 2.6

Height ← 10

Triangle ← (Base * Height) / 2

Radius ← 1

Height ← 2

Cone  ← 2 * Pi * Radius * (Radius + Height)

a ← 13

b ← 7

c ← 3

Answer1 ← NOT((a + b + c) > 28)

Total ← 34

Total ← Total - 2

Answer2 ← a + c * c
```

Give the final value assigned to each variable.

(i) `Triangle` ..... [1]

(ii) `Cone` ..... [1]

(iii) `Answer1` ..... [1]

(iv) `Total` ..... [1]

(v) `Answer2` ..... [1]

Appendix

Built-in functions (pseudocode)

```
ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
```

returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.

For example: `ONECHAR("New York", 5)` returns `'Y'`

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```

returns the number of characters in `ThisString`.

For example: `CHARACTERCOUNT("New York")` returns `8`

```
SUBSTR(ThisString : STRING, Value1 : INTEGER, Value2 : INTEGER) RETURNS
STRING
```

returns a sub-string from within `ThisString`.
`Value1` is the start index position (counting from the left, starting with 1).
`Value2` is the final index position.

For example: `SUBSTR("art nouveau", 5, 11)` returns `"nouveau"`

```
TONUM(ThisString : STRING) RETURNS INTEGER or REAL
```

returns the integer or real equivalent of the string `ThisString`.

For example: `TONUM("502")` returns the integer `502`
`TONUM("56.36")` returns the real number `56.36`

```
ASC(ThisCharacter : CHAR) RETURNS INTEGER
```

returns an integer which is the ASCII character code for the character `ThisCharacter`.

For example: `ASC('A')` returns integer `65`

```
CHR(Value : INTEGER) RETURNS CHAR
```

returns the character that ASCII code number `Value` represents.

For example: `CHR(65)` returns `'A'`

```
RND() RETURNS REAL
```

returns a random number in the range 0 to 0.99999

For example: `RND()` returns `0.67351`

```
INT(ThisNumber : REAL) RETURNS INTEGER
```

returns the integer part of `ThisNumber`.

For example: `INT(12.79)` returns `12`

Errors

For any function, if the program calls the function incorrectly, the function returns an error.

Concatenation operator

`&` – Concatenates two expressions of `STRING` or `CHAR` data type.

For example: `"South" & " " & "Pole"` produces `"South Pole"`
`'B' & "000654"` produces `"B000654"`

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2016

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2015**

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 **19** printed pages and **1** blank page.

DC (NF/JG) 112056/1
© UCLES 2015

CAMBRIDGE International Examinations

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

1 A marathon runner records their time for a race in hours, minutes and seconds.

An algorithm is shown below in structured English.

INPUT race time as hours, minutes and seconds
CALCULATE race time in seconds
STORE race time in seconds
OUTPUT race time in seconds

(a) The identifier table needs to show the variables required to write a program for this algorithm.

Complete the table.

Identifier	Data type	Description
`RaceHours`	`INTEGER`	The hours part of the race time.

[3]

(b) Before the program is written, the design is amended.

The new design includes input of the runner's current personal best marathon time (in seconds).

The output will now also show one of the following messages:

- "Personal best time is unchanged"
- "New personal best time"
- "Equals personal best time"

(i) Show the additional variable needed for the new design.

Identifier	Data type	Description

[1]

(ii) Write **program code** for the new design.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[7]

(c) The program code will be tested using white-box testing.

(i) Explain what is meant by white-box testing.

.....

.....

.....[2]

(ii) Complete the table heading.
Complete Test Number 1.
Add the data for Test Number 2 and Test Number 3.

	Input values				**Output**	
Test number	**Race hours**	**Race minutes**	**Race seconds**	**.....**	**Total time (seconds)**	**Message**
1	3	4	13	11053	11053	
2				11053		
3				11053		

[6]

© UCLES 2015

2 A program displays a menu with choices 1 to 4. The code to display the menu is written as the procedure `DisplayMenu`.

(a) Pseudocode which uses this procedure is:

```
CALL DisplayMenu
REPEAT
   OUTPUT "Enter choice (1..4)"
   INPUT Choice
UNTIL Choice >= 1 AND Choice <= 4
```

(i) Describe what this pseudocode will do.

.....

.....

.....

.....[3]

(ii) State why a loop is required.

.....

.....[1]

(b) The following pseudocode is a revised design.

```
CONSTANT i ← 3
CALL DisplayMenu
NoOfAttempts ← 0
REPEAT
   OUTPUT "Enter choice (1..4)"
   INPUT Choice
   NoOfAttempts ← NoOfAttempts + 1
UNTIL (Choice >= 1 AND Choice <= 4) OR NoOfAttempts = i
```

(i) Give the maximum number of inputs the user could be prompted to make.

..... [1]

(ii) State why this algorithm is an improvement on the one given in **part (a)**.

.....

.....[1]

(c) The pseudocode is in its initial stage of development.

The table below shows the action currently taken by the pseudocode following each menu choice.

Menu choice	Description	Program response
1	Read data from the customer file	Calls a procedure `ReadFile` which for testing purposes outputs the message “Read file code”
2	Add a customer	Outputs message “Add customer code”
3	Search for a customer	Outputs message “Search customer code”
4	Terminates the program	Ends

Complete the pseudocode for the design in **part (b)**, shown again below, to respond to each menu choice.

```
CONSTANT i ← 3
CALL DisplayMenu
NoOfAttempts ← 0
REPEAT
   OUTPUT "Enter choice (1..4)"
   INPUT Choice
   NoOfAttempts ← NoOfAttempts + 1
UNTIL (Choice >= 1 AND Choice <= 4) OR NoOfAttempts = i
```

.....

.....

.....

.....

.....

.....

.....[3]

(d) The algorithm in **part (c)** is to be amended. The program will:

- repeatedly display the menu and respond to the user's choice
- terminate when the user enters 4

Write **program code** for this final design which will be made up of:

- the main program
- procedure `ReadFile`
- procedure `DisplayMenu`

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

3 When the guarantee on a computer runs out, the owner can take out insurance to cover breakdown and repairs.

The price of the insurance is calculated from:

- the model of the computer
- the age of the computer
- the current insurance rates

Following an enquiry to the insurance company, the customer receives a quotation letter with the price of the insurance.

A program is to be produced.

The structure chart below shows the modular design for this process:

(a) Using the letters **A** to **D**, add the labelling to the chart boxes on the opposite page.

Modules	
A	Send quotation letter
B	Calculate price
C	Produce insurance quotation
D	Input computer details

[2]

(b) Using the letters **E** to **J**, complete the labelling on the chart opposite.

Some of these letters will be used more than once.

Data items	
E	`CustomerName`
F	`CustomerEmail`
G	`Model`
H	`Age`
I	`PolicyCharge`
J	`PolicyNumber`

[4]

4 A game is played between two players:

- they take turns at rolling a six-sided die (numbered 1 to 6) and record their throw
- a player scores 1 point if their throw is higher than their opponent
- they each roll the die 20 times
- if the player's throw is the same as their opponent, the total points is unchanged
- the winner is the player with the larger number of points after 20 throws

The pseudocode will use variable `NoOfThrows` as shown.

Identifier	Data type	Description
`NoOfThrows`	`INTEGER`	Loop control variable

(i) Complete the pseudocode for the given algorithm.

```
FOR ..............................................................................

  INPUT Player1Throw

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

  IF Player1Throw > Player2Throw
    THEN

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

  ENDIF
  IF Player2Throw > Player1Throw
    THEN
      Player2Total ← Player2Total + 1
  ENDIF

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

IF Player1Total > Player2Total
  THEN
    OUTPUT "Player1 is the winner"
  ELSE
    OUTPUT "Player2 is the winner"
ENDIF
```

[5]

(ii) Identify the game result which will produce incorrect output.

.....

.....[1]

Question 5 begins on page 12.

5 A company creates two new websites, Site X and Site Y, for selling bicycles.

Various programs are to be written to process the sales data.

These programs will use data about daily sales made from Site X (using variable `SalesX`) and Site Y (using variable `SalesY`).

Data for the first 28 days is shown below.

	`SalesDate`	`SalesX`	`SalesY`
1	03/06/2015	0	1
2	04/06/2015	1	2
3	05/06/2015	3	8
4	06/06/2015	0	0
5	07/06/2015	4	6
6	08/06/2015	4	4
7	09/06/2015	5	9
8	10/06/2015	11	9
9	11/06/2015	4	1
…			
28	01/07/2015	14	8

(a) Name the data structure to be used in a program for `SalesX`.

.....[2]

(b) The programmer writes a program from the following pseudocode design.

```
x ← 0
FOR DayNumber ← 1 TO 7
   IF SalesX[DayNumber] + SalesY[DayNumber] >= 10
      THEN
         x ← x + 1
         OUTPUT SalesDate[DayNumber]
   ENDIF
ENDFOR
OUTPUT x
```

(i) Trace the execution of this pseudocode by completing the trace table below.

x	DayNumber	OUTPUT
0		

[4]

(ii) Describe, in detail, what this algorithm does.

.....

.....

.....

.....[3]

(c) The company wants a program to output the total monthly sales for one of the selected websites.

The programmer codes a function with the following function header:

```
FUNCTION MonthlyWebSiteSales(ThisMonth : INTEGER, ThisSite : CHAR)
                                                    RETURNS INTEGER
```

The function returns the total number of bicycles sold for the given month and website.

The function will use the following:

Identifier	Data type	Description
`ThisMonth`	`INTEGER`	Represents the month number e.g. 4 represents April
`ThisSite`	`CHAR`	Coded as: • X for website X • Y for Website Y

(i) Give the number of parameters of this function. .....[1]

(ii) Some of the following function calls may be invalid.

Mark each call with:

- a tick (✓), for a valid call
- a cross (✗), for an invalid call

For any function calls which are invalid, explain why.

Function call	Tick (✓) / cross (✗)	Explanation (if invalid)
`MonthlyWebSiteSales(1, "Y")`		
`MonthlyWebSiteSales(11, 'X', 'Y')`		
`MonthlyWebSiteSales(12, 'X')`		

[3]

(d) The company decides to offer a discount on selected dates. A program is written to indicate the dates on which a discount is offered.

The program creates a text file, DISCOUNT_DATES (with data as shown), for a number of consecutive dates.

03/06/2015 TRUE
04/06/2015 FALSE
05/06/2015 FALSE
06/06/2015 FALSE
07/06/2015 FALSE
08/06/2015 FALSE
09/06/2015 FALSE
10/06/2015 TRUE
11/06/2015 FALSE
01/07/2015 FALSE

Each date and discount indicator is separated by a single <Space> character.

The discount indicators are:

- FALSE – indicates a date on which no discount is offered
- TRUE – indicates a date on which a discount is offered

A programming language has the built-in function `CONCAT` defined as follows:

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING] )
                                                    RETURNS STRING
```

For example:

```
CONCAT("San", "Francisco") returns "SanFrancisco"
CONCAT("New", "York", "City") returns "NewYorkCity"
```

The use of the square brackets indicates that the parameter is optional.

The following incomplete pseudocode creates the text file DISCOUNT_DATES.

Complete the pseudocode.

```
OPENFILE "DISCOUNT_DATES" FOR ..............................................................

INPUT ......................................................................................

WHILE  NextDate <>"XXX"

   INPUT Discount

   ..............................  = CONCAT(NextDate, " ", Discount)

   WRITEFILE "DISCOUNT_DATES",  NextLine

   INPUT NextDate

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

OUTPUT "File now created"

CLOSEFILE
```

[4]

Question 5(e) continues on page 18.

(e) The `DISCOUNT_DATES` text file is successfully created.

The company now wants a program to:

- key in a date entered by the user
- search the text file for this date
- if found, output one of the following messages:
  - "No discount on this date"
  - "This is a discount date"
- if not found, output "Date not found"

(i) Add to the identifier table to show the variables you need for this new program.

Identifier	Data type	Description
`DISCOUNT_DATES`	`FILE`	Text file to be used

[3]

(ii) Write the **program code**.

Do not include any declaration or comment statements for the variables used.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[7]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2015

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2021**

INSERT **2 hours**

3013807674-I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (MB) 200598/3
© UCLES 2021

Note: An error occurs if a function call is not properly formed, or if the parameters are incorrect.

STRING Functions

`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"`

`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`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the lower case equivalent of `ThisChar`
Characters that are not upper case alphabetic are returned unchanged

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

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the upper case equivalent of `ThisChar`
Characters that are not lower case alphabetic are returned unchanged

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

`TO_UPPER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to upper case

Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`

`TO_LOWER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to lower case

Example: `TO_LOWER("JIM 803")` returns `"jim 803"`

`NUM_TO_STR(x : <data type>) RETURNS STRING`
returns a string representation of a numeric value
Note: `<data type>` may be `REAL` or `INTEGER`

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

```
STR_TO_NUM(x : <data type1>) RETURNS <data type2>
```
returns a numeric representation of a string
Note: `<data type1>` may be `CHAR` or `STRING`
Note: `<data type2>` may be `REAL` or `INTEGER`

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

```
IS_NUM(ThisString : STRING) RETURNS BOOLEAN
```
returns the value `TRUE` if `ThisString` represents a valid numeric value
Note: `<data type>` may be `CHAR` or `STRING`

Example: `IS_NUM("12.36")` returns `TRUE`
Example: `IS_NUM("-12.36")` returns `TRUE`
Example: `IS_NUM("12.3a")` returns `FALSE`

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

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

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

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

NUMERIC Functions

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

Example: `INT(27.5415)` returns `27`

```
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`

DATE Functions

Note: Date format is assumed to be DDMMYYYY unless otherwise stated.

```
DAY(ThisDate : DATE) RETURNS INTEGER
```
returns the current day number from `ThisDate`

Example: `DAY(4/10/2003)` returns `4`

```
MONTH(ThisDate : DATE) RETURNS INTEGER
```
returns the current month number from `ThisDate`

Example: `MONTH(4/10/2003)` returns `10`

`YEAR(ThisDate : DATE) RETURNS INTEGER`
returns the current year number from `ThisDate`

Example: `YEAR(4/10/2003)` returns `2003`

`DAYINDEX(ThisDate : DATE) RETURNS INTEGER`
returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2, Tuesday = 3 etc.

Example: `DAYINDEX(12/05/2020)` returns `3`

`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE`
returns a variable of type `DATE`

`NOW() RETURNS DATE`
returns the current date

OTHER Functions

`EOF(FileName : STRING) RETURNS BOOLEAN`
returns `TRUE` if there are no more lines to be read from file `FileName`

Note: This function will generate an ERROR if the file is not already open in `READ` mode

OPERATORS

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: This operator may also be used to concatenate a character with a string
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example `10 DIV 3` evaluates to `3`

Note: An error is generated if an operator is used with a value or values of an incorrect type.

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

4860204797

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2020**

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 **20** pages. Blank pages are indicated.

DC (RW/TP) 180777/4
© UCLES 2020

BLANK PAGE

1 **(a)** Selection and repetition are basic constructs of an algorithm.

Name **and** describe **one other** construct.

Name .....

Description .....

.....

.....
[3]

(b) Program coding is a transferable skill.

Explain the term **transferable skill**.

.....

.....

.....

..... [2]

(c) Count-controlled and post-condition are two types of loop.

Describe the characteristics of each of these types of loop.

Count-controlled .....

.....

.....

Post-condition .....

.....

.....
[2]

(d) Name **three** features provided by an Integrated Development Environment (IDE) that assist in the coding and **initial** error detection stages of the program development cycle.

1 .....

2 .....

3 .....
[3]

2 **(a)** A structure chart is often produced as part of a modular program design. The chart shows the hierarchy of modules and the sequence of execution.

Give **two other** features the structure chart can show.

Feature 1 .....

.....

Feature 2 .....

.....
[2]

(b) Six program modules implement part of an online shopping program. The following table gives the modules and a brief description of each module:

Module	Description
`Shop()`	Allows the user to choose a delivery slot, select items to be added to the basket and finally check out
`ChooseSlot()`	Allows the user to select a delivery time. Returns a delivery slot number
`FillBasket()`	Allows the user to select items and add them to the basket
`Checkout()`	Completes the order by allowing the user to pay for the items. Returns a Boolean value to indicate whether or not payment was successful
`Search()`	Allows the user to search for a specific item. Returns an item reference
`Add()`	Adds an item to the basket. Takes an item reference and a quantity as parameters

(i) The online shopping program has been split into sub-tasks as part of the design process.

Explain the advantages of decomposing the program into modules. Your explanation should refer to the scenario and modules described in **part (b)**.

.....

.....

.....

.....

.....

..... [3]

(ii) Complete the structure chart for the six modules described in **part (b)**.

[6]

3 A navigation program includes a function, `CheckCourse()`. This function is called with a real value, `Course`, and returns an integer value.

The identifier table and the program flowchart for the function are shown as follows:

Identifier	Type	Description
`Course`	`REAL`	The value passed to `CheckCourse()`
`Adjust`	`INTEGER`	The value returned by `CheckCourse()`
`Check`	`INTEGER`	A local variable
`Deviate()`	`FUNCTION`	A function that is passed a `REAL` value representing the course and returns a `REAL` value representing the current deviation
`Alert()`	`PROCEDURE`	A procedure that generates a warning

Write **pseudocode** to implement the function `CheckCourse()`. The pseudocode must follow the algorithm represented by the flowchart. Declare any local variables used.

Refer to the **Appendix** on page 19 for a list of built-in pseudocode functions and operators.

..... [7]

4 **(a)** Mustafa has developed the following pseudocode to generate ten random integers in the range 1 to 100.

```
DECLARE Random : ARRAY [1:10] OF INTEGER
DECLARE Count, RNum : INTEGER

FOR Count ← 1 TO 10
   Rnum ← INT(RAND(100)) + 1
   Random[Count] ← RNum
ENDFOR
```

Refer to the **Appendix** on page 19 for a list of built-in pseudocode functions and operators.

Rewrite the pseudocode so that there are no duplicated numbers in the list of random numbers.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) The changes made to the pseudocode in **part (a)** were as a result of changes to the program requirement.

Give the term used to describe changes made for this reason.

..... [1]

5 A global 1D array, `Contact`, of type `STRING` is used to store a list of names and email addresses.

There are 1000 elements in the array. Each element stores one data item. The format of each data item is as follows:

```
<Name>':'<EmailAddress>
```

`Name` and `EmailAddress` are both variable-length strings.

For example:

```
"Sharma Himal:hsharma99@stlmail.com"
```

A function, `GetName()`, is part of the program that processes the array. A data item string will be passed to the function as a parameter. The function will return the `Name` part. Validation is **not** necessary.

(a) Use **structured English** to describe the algorithm for the function `GetName()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) **(i)** The array is to be sorted using an efficient bubble sort algorithm. An efficient bubble sort reduces the number of unnecessary comparisons between elements.

Describe how this could be achieved.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(ii) A procedure, `BubbleSort()`, is needed to sort the 1D array `Contact` into ascending order of `Name` using an efficient bubble sort algorithm.

Write **program code** for the procedure `BubbleSort()`.

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]

Question 6 begins on the next page.

6 A company hires out rowing boats on a lake. The company has 17 boats numbered from 1 to 17.

Boats may be hired between 9:00 and 18:00, with a maximum hire duration of 90 minutes.

The company is developing a program to help manage and record the boat hire process.

The programmer has decided to store all values relating to hire time as strings. The program will use a 24-hour clock format. For example:

Time (in words)	String value
Nine o'clock in the morning	`"09:00"`
Five minutes past ten o'clock in the morning	`"10:05"`
Ten minutes before three o'clock in the afternoon	`"14:50"`

The programmer has defined the first module as follows:

Module	Description
`AddTime()`	• Takes two parameters: ◦ `StartTime`: a `STRING` value representing a time as described ◦ `Duration`: an `INTEGER` value representing a duration in minutes • Adds the duration to the time to give a new time • Returns the new time as a `STRING`

(a) (i) Write **pseudocode** for the module `AddTime()`. Assume both input parameters are valid.

Refer to the **Appendix** on page 19 for a built-in list of pseudocode functions and operators.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(ii) `AddTime()` will be tested using white-box testing.

State the reason for using white-box testing.

.....

..... [1]

(iii) A run-time error is one type of error that black-box testing can reveal.

Describe **one other** type of error that black-box testing can reveal.

.....

..... [2]

(b) The user will input the desired start time of a hire. A new module will be written to validate the input string as a valid time in 24-hour clock format.

The string is already confirmed as being in the format `"NN:NN"`, where `N` is a numeric character.

Give an example of suitable test data that is in this format but which is **invalid**. Explain your answer.

Test data .....

Explanation .....

.....

.....

.....
[2]

(c) Each time a boat is hired out, details of the hire are added to a text file, `Hirelog.txt`. Each line of the text file corresponds to information about one hire session.

The format of each line is as follows:

```
<BoatNumber><Date><AmountPaid>
```

- `BoatNumber` is a two-digit numeric string
- `Date` is a six-digit numeric string in the format DDMMYY
- `AmountPaid` is a variable-length string representing a numeric value, for example `"12.75"`

The total hire amount from each boat is to be stored in a global array, `Total`. This array is declared in pseudocode as follows:

```
DECLARE Total : ARRAY [1:17] OF REAL
```

The programmer has defined module `GetTotals()` as follows:

Module	Description
`GetTotals()`	• Search through the file `Hirelog.txt` • Extract the `AmountPaid` each time a boat is hired • Store the total of `AmountPaid` for each boat in the array

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

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]

BLANK PAGE

Appendix

Built-in functions (pseudocode)

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

`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`

`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`

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

`DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`

Example: `DIV(10,3)` returns `3`

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

Example: If `x` has the value `87.5` then `NUM_TO_STRING(x)` returns `"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: If `x` has the value `"23.45"` then `STRING_TO_NUM(x)` returns `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`

© UCLES 2020

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.

© UCLES 2020

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 9 9 3 0 7 9 3 4 0 8 *

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2016**

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 **20** printed pages.

DC (RW/FD) 115865/3
© UCLES 2016

CAMBRIDGE International Examinations

There is an **Appendix** on pages 19 and 20. Some questions will refer you to this information.

1 A programmer wants to write a program to calculate the baggage charge for a passenger's airline flight.

Two types of ticket are available for a flight:

- economy class (coded E)
- standard class (coded S)

Each ticket type has a baggage weight allowance as shown below. The airline makes a charge if the weight exceeds the allowance.

Ticket type	Baggage allowance (kg)	Charge rate per additional kg ($)
'E'	16	3.50
'S'	20	5.75

(a) A program flowchart will document the program. The flowchart will contain the following statements:

Statement number	Statement
1	`Charge ← 0`
2	`INPUT BaggageWeight`
3	`Charge ← ExcessWeight * ChargeRate`
4	`Is ExcessWeight > 0 ?`
5	`INPUT TicketType`
6	`ExcessWeight ← BaggageWeight - BaggageAllowance`
7	`BaggageAllowance ← 16`
8	`ChargeRate ← 3.5`
9	`OUTPUT Charge`
10	`ChargeRate ← 5.75`
11	`BaggageAllowance ← 20`
12	`Is TicketType = 'E' ?`

Complete the flowchart by putting the appropriate **statement number** in each flowchart symbol. Statement 5 has been done for you.

[6]

(b) The programmer needs data to test the flowchart.

Complete the table of test data below to show **five** tests.

`TicketType`	`BaggageWeight`	Explanation	Expected output
`E`	`15`	....	
		....	
		....	
		....	
		....	

[5]

(c) The program design is to be amended. The value input by the user for the ticket type is to be validated. Part of the amended flowchart is shown below.

Write **pseudocode** to use a pre-condition loop for this validation.

.....

.....

.....

.....

.....

.....[3]

2 A sensing device sends bit values to a computer along data channels.

- Channel 1 transmits a sequence of binary values from a sensor
- Channel 2 transmits at regular intervals to indicate whether the sensor is switched on or off:
  - 0 indicates switched off
  - 1 indicates switched on

A program tests the bits received from the sensing device.

A program reads the signal from Channel 2 after every six values from Channel 1.

A built-in function `READ(<ChannelNumber>)` reads a value from the specified channel.

Pseudocode for the program is as follows:

```
BitCount ← 0
Status2 ← READ(2)
WHILE Status2 = 1

  FOR ReadingCount ← 1 TO 6
    ThisBit ← READ(1)
    IF ThisBit = 1
      THEN
        BitCount ← BitCount + 1
    ENDIF
    IF BitCount = 5
      THEN
        OUTPUT "Error - Investigate"
        BitCount ← 0
    ENDIF
  ENDFOR

  Status2 ← READ(2)
ENDWHILE
```

(a) Trace the execution of the program for the following sequence of bits.

Channel 1		1	0	1	1	1	0		1	1	0	0	1	1	
Channel 2	1							1							0

Status2	ReadingCount	ThisBit	BitCount	OUTPUT
			0	
1	1	1	1	
	2			

[7]

(b) Identify the following constructs in the given program, using line numbers.

For multi-line constructs give the first line number only.

Construct	Line number
Assignment	
Selection	
Iteration	

[3]

3 You will need to refer to the list of pseudocode string-handling functions in the **Appendix**.

ASCII code table (part)					
Character	Decimal	Character	Decimal	Character	Decimal
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

(a) For each statement, write the value assigned to the variable.

(i) `Term ← CHARACTERCOUNT("TSUNAMI")`

`Term` .....[1]

(ii) `Answer1 ← ASC('G') + ASC(<Space>)`

`Answer1` .....[1]

(iii) `Answer2 ← CHR(CHARACTERCOUNT("HELLO") + 70)`

`Answer2` .....[1]

(iv) `Word ← SUBSTR("Welcome home", 4, 7) )`

`Word` .....[1]

Question 3(b) continues on page 10.

(b) A programmer wants to design a procedure to calculate a customer ID number from the customer's surname.

The procedure will:

- input the surname
- isolate each character in the surname and find the corresponding ASCII code
- calculate the total of all these ASCII codes
- this total is the customer ID

(i) Complete the pseudocode for this procedure.

You will need to refer to the list of pseudocode string-handling functions in the Appendix.

```
PROCEDURE CalculateCustomerID

  OUTPUT "Key in surname"

  INPUT Surname

  Length ← ..................................................................................

  CustomerID ← 0

  FOR i ← 1 TO Length

    // NextChar is a single character from Surname

    NextChar ← ..............................................................................

    NextCodeNumber ← ASC(NextChar)

    CustomerID ← CustomerID + ...............................................................

  ENDFOR

  OUTPUT "Customer ID is ", CustomerID
```

[3]

(ii) Write **program code** for procedure `CalculateCustomerID`.

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

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

(c) The programmer decides that it would be better to write the procedure as a function. The user will now input the surname in the main program.

Write **program code** for the following:

State your programming language .....

(i) The function header for this new function `CalculateCustomerID`

.....[3]

(ii) The additional statement required within the function body to complete the change from a procedure to a function.

.....

.....[1]

(iii) The statement in the main program which:

- calls the function for surname Wilkes
- assigns the result to variable `ThisID`

.....[3]

(d) **(i)** The new function `CalculateUserID` is an example of a 'user-defined function'.

State **two** differences between a built-in function and a user-defined function.

1 .....

.....

2 .....

.....[2]

(ii) State **two** things that built-in and user-defined functions have in common.

1 .....

.....

2 .....

.....[2]

4 A company employs Ahmed as a programmer.

(a) At College, before joining the company, Ahmed used two items of software for programming:

- a text editor
- a compiler

Describe how he could have developed programs using these software tools.

Include in the description the terms 'object code' and 'source code'.

.....

.....

.....

.....

.....

.....

.....[3]

(b) Ahmed now uses an Integrated Development Environment (IDE) for programming.

(i) State **one** feature an IDE provides to help with the identification of syntax errors.

.....

.....[1]

(ii) State **one** feature an IDE provides to carry out white box testing.

.....

.....[1]

(c) The company maintains a file of product data. Ahmed is to write a program to add a new product and search for a product based on the structure diagram shown:

The program records the following data for each product:

- product code
- product description
- product retail price

The text file `PRODUCTS` stores each data item on a separate line, as shown below:

File **PRODUCTS**

0198
Plums(10kg)
11.50
0202
Onions(20kg)
10.00
~
0376
Mango chutney(1kg)
02.99
~
0014
Mango(10kg)
12.75

The program uses the variables shown in the identifier table.

Identifier	Data type	Description
PRODUCTS	TEXT FILE	Storing the code, description and retail price for all current products
PCode	ARRAY[1:1000] OF STRING	Array storing the product codes
PDescription	ARRAY[1:1000] OF STRING	Array storing the product descriptions
PRetailPrice	ARRAY[1:1000] OF REAL	Array storing the product retail prices
i	INTEGER	Array index used by all three arrays

© UCLES 2016

(i) The first operation of the program is to read all the product data held in file `PRODUCTS` and write them into the three 1D arrays.

Complete the pseudocode below.

```
OPEN ..........................................................................

i ← 1

WHILE ..........................................................................

  READFILE ("PRODUCTS", ........................................................)

  READFILE ("PRODUCTS", ........................................................)

  READFILE ("PRODUCTS", ........................................................)

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

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

ENDWHILE

CLOSE "PRODUCTS"

OUTPUT "Product file contents written to arrays"
```

[5]

When Ahmed designed the `PRODUCTS` file, he considered the alternative file structure shown opposite.

It stores one product per line in the text file.

File **PRODUCTS**

0198 Plums(10kg) 11.50
0202 Onions(20kg) 10.00
⋮
0376 Mango chutney(1kg) 02.99
⋮
0014 Mango(10kg) 12.75

(ii) State **one** benefit and **one** drawback of this file design.

Benefit .....

.....

Drawback .....

.....[2]

(d) To code the 'Search by product code' procedure, Ahmed draws a structure chart showing the different stages.

The procedure uses the variables shown in the identifier table.

Identifier	Data type	Description
`SearchCode`	`STRING`	Product code input by the user
`ThisIndex`	`INTEGER`	Array index position for the corresponding product
`ThisDescription`	`STRING`	Product description found
`ThisRetailPrice`	`REAL`	Product retail price found

You can assume that before the procedure is run, all the product data is read from file `PRODUCTS` and then stored in three 1D arrays as described in **part (c)(i)**.

Label the structure chart to show the input(s) and output(s).

[4]

(e) A first attempt was made at writing the 'Search for product code' module.
Ahmed designs this as a function `ProductCodeSearch`.

The function returns an integer value as follows:

- if the product code is found, it returns the index position of the 1D array `PCode` being searched
- if the product code is not found, the function returns -1

Write **program code** for function `ProductCodeSearch`.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

5 Study the following pseudocode statements.

```
CONST  Pi =  3.1          : REAL

DECLARE Triangle, Base, Height, Radius, Cone : REAL

DECLARE a, b, c, Answer2 : INTEGER

DECLARE Answer1           : BOOLEAN

Base ← 2.6

Height ← 10

Triangle ← (Base * Height) / 2

Radius ← 1

Height ← 2

Cone  ← 2 * Pi * Radius * (Radius + Height)

a ← 13

b ← 7

c ← 3

Answer1 ← NOT((a + b + c) > 28)

Total ← 34

Total ← Total - 2

Answer2 ← a + c * c
```

Give the final value assigned to each variable.

(i) `Triangle` ..... [1]

(ii) `Cone` ..... [1]

(iii) `Answer1` ..... [1]

(iv) `Total` ..... [1]

(v) `Answer2` ..... [1]

Appendix

Built-in functions (pseudocode)

```
ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
```

returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.

For example: `ONECHAR("New York", 5)` returns `'Y'`

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```

returns the number of characters in `ThisString`.

For example: `CHARACTERCOUNT("New York")` returns `8`

```
SUBSTR(ThisString : STRING, Value1 : INTEGER, Value2 : INTEGER) RETURNS
STRING
```

returns a sub-string from within `ThisString`.
`Value1` is the start index position (counting from the left, starting with 1).
`Value2` is the final index position.

For example: `SUBSTR("art nouveau", 5, 11)` returns `"nouveau"`

```
TONUM(ThisString : STRING) RETURNS INTEGER or REAL
```

returns the integer or real equivalent of the string `ThisString`.

For example: `TONUM("502")` returns the integer `502`
`TONUM("56.36")` returns the real number `56.36`

```
ASC(ThisCharacter : CHAR) RETURNS INTEGER
```

returns an integer which is the ASCII character code for the character `ThisCharacter`.

For example: `ASC('A')` returns integer `65`

```
CHR(Value : INTEGER) RETURNS CHAR
```

returns the character that ASCII code number `Value` represents.

For example: `CHR(65)` returns `'A'`

```
RND() RETURNS REAL
```

returns a random number in the range 0 to 0.99999

For example: `RND()` returns `0.67351`

```
INT(ThisNumber : REAL) RETURNS INTEGER
```

returns the integer part of `ThisNumber`.

For example: `INT(12.79)` returns `12`

Errors

For any function, if the program calls the function incorrectly, the function returns an error.

Concatenation operator

`&` – Concatenates two expressions of `STRING` or `CHAR` data type.

For example: `"South" & " " & "Pole"` produces `"South Pole"`
`'B' & "000654"` produces `"B000654"`

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2016

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

0113520592

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2023**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (LK/SG) 315866/3
© UCLES 2023

Refer to the **insert** for the list of pseudocode functions and operators.

1 The following pseudocode represents part of the algorithm for a program:

```
CASE OF ThisValue
   < 30       : Level ← "Low"           // less than 30
                Check ← 1
   < 20       : Level ← "Very Low"      // less than 20
                Check ← ThisValue / 2
   30 TO 40   : Level ← "Medium"        // between 30 and 40
                Check ← ThisValue / 3
                Data[ThisValue] ← Data[ThisValue] + 1
   > 40       : Level ← "High"
ENDCASE
```

(a) Complete the table by writing the answer for each row:

	Answer
The value assigned to `Level` when `ThisValue` is 40	
The value assigned to `Check` when `ThisValue` is 36	
The value assigned to `Level` when `ThisValue` is 18	
The number of elements in array `Data` that may be incremented	

[4]

(b) The pseudocode contains four assignments to variable `Level`. One of these assignments will never be performed.

Identify this assignment **and** explain why this is the case.

.....

.....

.....

.....

.....

..... [3]

(c) The following line is added immediately before the `ENDCASE` statement:

```
OTHERWISE : Level ← "Undefined"
```

State why this assignment is never performed.

.....

..... [1]

(d) Give the appropriate data types for the variables `ThisValue`, `Check` and `Level`.

`ThisValue` .....

`Check` .....

`Level` .....

[3]

2 **(a)** An algorithm is expressed as follows:

- input 100 numbers, one at a time
- keep a total of all numbers input that have a value between 30 and 70 inclusive and output this total after the last number has been input.

Outline, using stepwise refinement, the five steps for this algorithm which could be used to produce pseudocode.

Do **not** use pseudocode statements in your answer.

Step 1 .....

.....

Step 2 .....

.....

Step 3 .....

.....

Step 4 .....

.....

Step 5 .....

.....

[5]

(b) Sequence is one programming construct.

Identify **two other** programming constructs that will be required when the algorithm is converted into pseudocode.

Construct 1 .....

.....

Construct 2 .....

.....

[2]

3 The diagram represents an Abstract Data Type (ADT).

The operation of this stack may be summarised as follows:

- The `TopOfStack` pointer points to the last item added to the stack.
- The `BottomOfStack` pointer points to the first item on the stack.

Stack

D1	⟵ `TopOfStack`
D3	
D4	
D5	
D2	⟵ `BottomOfStack`

(a) The stack is implemented using two variables and a 1D array of 8 elements as shown.

The variables are used to reference individual elements of the array, in such a way that:

- the array is filled from the lowest indexed element towards the highest
- all the elements of the array are available for the stack.

Complete the diagram to represent the state of the stack as shown above.

Array element	Data
8	
7	
6	
5	
4	
3	
2	
1	

Variable	
`TopOfStack`	
`BottomOfStack`	

[3]

(b) A function `Push()` will add a value onto the stack by manipulating the array and variables in **part (a)**.

Before adding a value onto the stack, the algorithm will check that space is available.

If the value is added to the stack, the function will return `TRUE`, otherwise it will return `FALSE`.

The algorithm is expressed in five steps.

Complete the steps.

1. If ..... then return `FALSE`
2. Otherwise ..... `TopOfStack`
3. Use `TopOfStack` as an ..... to the array.
4. Set the element at this ..... to the ..... being added.
5. Return ..... .

[5]

4 A global array is declared in pseudocode as follows:

```
DECLARE Data : ARRAY[1:150] OF STRING
```

A function `TooMany()` will:

1. take two parameters:
   - a string (the search string)
   - an integer (the maximum value)
2. count the number of strings in the array that exactly match the search string
3. return `TRUE` if the count is greater than the maximum value, otherwise will return `FALSE`

(a) Write pseudocode for the function `TooMany()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) The global array is changed to a 2D array, organised as 150 rows by 2 columns. It is declared in pseudocode as follows:

```
DECLARE Data : ARRAY[1:150, 1:2] OF STRING
```

The algorithm for the function in **part (a)** is changed. Strings will only be counted if **both** of the following conditions are true:

- The current row is an even number.
- The search string exactly matches the value in **either** column.

Write pseudocode to check these conditions.

Assume that the row index is contained in variable `Row` and the search string in variable `Search`.

.....

.....

.....

.....

.....

..... [3]

5 An algorithm is designed to find the smallest numeric value from an input sequence and count how many numeric values have been input.
An example of an input sequence is:

23, AB56, 17, 23ZW, 4, 10, END

Numeric input values are all integers and non-numeric input is ignored, except for the string "END" which is used to terminate the sequence.

The algorithm is expressed in pseudocode as shown:

```
DECLARE NextInput : STRING
DECLARE Min, Count, Num : INTEGER

Min ← 999
Count ← 0

REPEAT
   INPUT NextInput
   IF IS_NUM(NextInput) = TRUE THEN
      Num ← STR_TO_NUM(NextInput)
      IF Num > Min THEN
         Min ← Num
      ENDIF
      Count ← Count & 1
   ENDIF
UNTIL NextInput ← "END"

OUTPUT "The minimum value is ", Min, " and the count was ", Count
```

(a) The pseudocode contains three errors due to the incorrect use of **operators**.

Identify each error **and** state the correction required.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(b) The operator errors are corrected and the algorithm is tested as follows:

The input sequence:

```
18, 4, ONE, 27, 189, ERIC, 3, 65, END
```

produces the output:

```
The minimum value is 3 and the count was 6
```

The algorithm is tested with a different test data sequence. The sequence contains a mix of integer and non-numeric values. It is terminated correctly but the algorithm produces unexpected results.

(i) Explain the problem with the algorithm.

.....

.....

.....

..... [2]

(ii) Give a sequence of **four** test data values that could be input to demonstrate the problem.

Value 1 .....

Value 2 .....

Value 3 .....

Value 4 .....
[2]

6 The pseudocode `OUTPUT` command starts each output on a new line.

(a) A new procedure `MyOutput()` will take a string and a Boolean parameter.
`MyOutput()` may be called repeatedly and will use concatenation to build a string using a global variable `MyString`, up to a maximum length of 255 characters.

`MyString` will be output in either of these two cases:

1. The Boolean parameter value is `TRUE`
2. The resulting string (after concatenation) would be longer than 255 characters.

If `MyString` is not output, the string is concatenated with `MyString`.

For example, the calls to `MyOutput()` given below would result in the output as shown:

```
MyOutput("Hello ", FALSE)
MyOutput("ginger ", FALSE)
MyOutput("cat", TRUE)
MyOutput("How are you?", TRUE)
```

Resulting output:

```
Hello ginger cat
How are you?
```

Notes:

- `MyString` is initialised to an empty string before `MyOutput()` is called for the first time.
- No string passed to `MyOutput()` will be longer than 255 characters.

Write pseudocode for `MyOutput()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) The design of the procedure given in **part (a)** is modified and `MyString` is changed from a global to a local variable declared in `MyOutput()`.

When the modified procedure is converted into program code, it does not work as expected.

Explain why it does not work as expected.

.....

.....

.....

..... [2]

7 An algorithm is represented by a state-transition diagram.

The table shows the inputs, outputs and states for the algorithm:

Current state	Input	Output	Next state
S1	A1	X1	S2
S2	A3	none	S1
S2	A2	X4	S5
S5	A1	X1	S5
S5	A4	X2	S2
S5	A3	none	S3
S1	A9	X9	S3
S3	A9	X9	S4

Complete the state-transition diagram to represent the information given in the table.

[5]

BLANK PAGE

8 A class of students are developing a program to send data between computers. Many computers are connected together to form a wired network. Serial ports are used to connect one computer to another.

Each computer:

- is assigned a unique three-digit ID
- has three ports, each identified by an integer value
- is connected to between one and three other computers.

Messages are sent between computers as a string of characters organised into fields as shown:

```
<STX><DestinationID><SourceID><Data><ETX>
```

Field number	Field name	Description
n/a	`STX`	a single character marking the start of the message (ASCII value 02)
1	`DestinationID`	three numeric characters that identify the destination computer
2	`SourceID`	three numeric characters that identify the source computer
3	`Data`	a variable length string containing the data being sent (Minimum length is 1 character)
n/a	`ETX`	a single character marking the end of the message (ASCII value 03)

For example, the following message contains the data "Hello Kevin" being sent from computer "101" to computer "232":

```
<STX>"232101Hello Kevin"<ETX>
```

Each computer will run a copy of the same program. Each program will contain a global variable, `MyID` of type string, that contains the unique ID of the computer in which the program is running.

The programmer has defined the first two program modules as follows:

Module	Description

`Transmit()` (already written)	• takes two parameters:
	  ○ a string containing a message
	  ○ an integer containing a port number
	• transmits the message using the given port

`SendFile()`	• takes three parameters:
	  ○ a string containing a text file name
	  ○ a string containing a Destination ID
	  ○ an integer containing a Port number
	• transmits the file one line at a time
	• transmits a final message with data string "`****`"

(a) Write pseudocode for module `SendFile()`.

Assume:

- module `Transmit()` has already been written and is used to transmit a message
- the value of `MyID` may be used as `SourceID`
- the file specified contains no blank lines
- the file specified does not contain the line "`****`"

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) Module `SendFile()` is used to copy a file from one computer to another.

A module within the program running on the destination computer will receive the data and write it to a new file.

Explain why module `SendFile()` transmits the message with data string `"****"` after the last line of the file.

.....

.....

.....

..... [2]

(c) One of the text files to be sent contains several blank lines (lines that do not contain any text).

(i) Explain why this is a problem.

.....

.....

.....

..... [2]

(ii) Explain how the message format could be changed to allow a blank line to be sent.

.....

.....

.....

..... [2]

BLANK PAGE

Question 8(d) starts on page 18.

(d) A new module has been defined:

Module	Description
`GetField()`	• takes two parameters: ○ a string containing a message ○ an integer containing a field number • If the field number is valid (in the range 1 to 3, inclusive), it returns a string containing the required field, otherwise it returns an empty string.

As a reminder, a message is defined as follows:

```
<STX><DestinationID><SourceID><Data><ETX>
```

Field number	Field name	Description
Not applicable	`STX`	a single character marking the start of the message (ASCII value 02)
1	`DestinationID`	three numeric characters that identify the destination computer
2	`SourceID`	three numeric characters that identify the source computer
3	`Data`	a variable length string containing the data being sent (Minimum length is 1 character)
Not applicable	`ETX`	a single character marking the end of the message (ASCII value 03)

Write pseudocode for module `GetField()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

0832678089

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2022**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (PQ/CGW) 303773/3
© UCLES 2022

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** The following table contains pseudocode examples.

Each example may include all or part of:

- selection
- iteration (repetition)
- assignment.

Complete the table by placing **one or more** ticks (✓) in each row.

Pseudocode example	Selection	Iteration	Assignment

`FOR Index ← 1 TO 3`
`  Safe[Index] ← GetResult()`
`NEXT Index`			

`OTHERWISE : OUTPUT "ERROR 1202"`			
`REPEAT UNTIL Index = 27`			
`INPUT MyName`			

`IF Mark > 74 THEN`
`  Grade ← 'A'`
`ENDIF`			

[5]

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

Variable	Value
AAA	TRUE
BBB	FALSE
Count	99

Complete the table by evaluating each expression.

Expression	Evaluation
`AAA AND (Count > 99)`	
`AAA AND (NOT BBB)`	
`(Count <= 99) AND (AAA OR BBB)`	
`(BBB AND Count > 50) OR NOT AAA`	

[2]

(ii) Give an example of when a variable of type Boolean would be used.

.....

..... [1]

2 A program has been written to implement a website browser and maintenance is now required.

One type of maintenance is called perfective.

Name **two other** types of maintenance that the program may require **and** give a reason for each.

Type 1 .....

Reason .....

.....

.....

.....

Type 2 .....

Reason .....

.....

.....

.....

[4]

3 Four program modules are defined as follows:

Pseudocode module header
`PROCEDURE Sub1_A(XT : INTEGER, PB : STRING)`
`FUNCTION Sub1_B(RA : INTEGER) RETURNS BOOLEAN`
`PROCEDURE Sub1_C(SB : INTEGER, BYREF SA : STRING)`
`PROCEDURE Section_1()`

(a) A structure chart will be produced as part of the development process.

Describe the purpose of a structure chart.

.....

.....

.....

..... [2]

(b) Module `Section_1()` calls one of the other three modules. The module called will be selected when the program runs.

Draw the structure chart.

[5]

4 Items in a factory are weighed automatically. The weight is stored as an integer value representing the item weight to the nearest gram (g).

A function is written to validate the weight of each item. The function will return "PASS" if the weight of the item is within the acceptable range, otherwise the function will return "FAIL".

The acceptable weight range for an item is 150 g to 155 g inclusive.

The validation function is to be properly tested. Black-box testing will be used and a test plan needs to be produced.

Complete the table by writing additional tests to test this function.

Type of test data	Example test value	Expected return value	Explanation
Normal	153	"PASS"	Value within the acceptable range

[4]

5 A program will store attendance data about each employee of a company.

The data will be held in a record structure of type `Employee`. The fields that will be needed are as shown:

Field	Typical value	Comment
`EmployeeNumber`	`123`	A numeric value starting from 1
`Name`	`"Smith,Eric"`	Format: <last name>','<first name>
`Department`	`"1B"`	May contain letters and numbers
`Born`	`13/02/2006`	Must not be before 04/02/1957
`Attendance`	`97.40`	Represents a percentage

(a) **(i)** Write pseudocode to declare the record structure for type `Employee`.

.....

.....

.....

.....

.....

.....

..... [4]

(ii) A 1D array `Staff` containing 500 elements will be used to store the employee records.

Write pseudocode to declare the `Staff` array.

.....

..... [2]

(b) There may be more records in the array than there are employees in the company. In this case, some records of the array will be unused.

(i) State why it is good practice to have a standard way to indicate unused array elements.

.....

..... [1]

(ii) Give **one** way of indicating an unused record in the `Staff` array.

.....

..... [1]

(c) A procedure `Absentees()` will output the `EmployeeNumber` and the `Name` of all employees who have an `Attendance` value of 90.00 or less.

Write pseudocode for the procedure `Absentees()`.

Assume that the `Staff` array is global.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

6 **(a)** The factorial of a number is the product of all the integers from 1 to that number.

For example:

factorial of 5 is given by $1 \times 2 \times 3 \times 4 \times 5 = 120$
factorial of 7 is given by $1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 5040$
factorial of 1 = 1

Note: factorial of 0 = 1

A function `Factorial()` will:

- be called with an integer number as a parameter
- calculate and return the factorial of the number
- return −1 if the number is negative.

Write pseudocode for the function `Factorial()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) A procedure `FirstTen()` will output the factorial of the numbers from 0 to 9. The procedure will use the function from **part (a)**.

The required output is:

```
Factorial of 0 is 1
Factorial of 1 is 1
Factorial of 2 is 2
   ⋮
Factorial of 9 is 362880
```

The program flowchart represents an algorithm for `FirstTen()`.

Complete the table by writing the text that should replace each label **A** to **F**.

Label	Text
A	
B	
C	
D	
E	
F	

[4]

7 The following pseudocode represents an algorithm intended to output the last three lines as they appear in a text file. Line numbers are provided for reference only.

```
10  PROCEDURE LastLines(ThisFile : STRING)
11     DECLARE ThisLine : STRING
12     DECLARE Buffer : ARRAY[1:3] OF STRING
13     DECLARE LineNum : INTEGER
14     LineNum ← 1
15     OPENFILE ThisFile FOR READ
16
17     WHILE NOT EOF(ThisFile)
18        READFILE Thisfile, ThisLine // read a line
19        Buffer[LineNum] ← ThisLine
20        LineNum ← LineNum + 1
21        IF LineNum = 4 THEN
22           LineNum ← 1
23        ENDIF
24     ENDWHILE
25
26     CLOSEFILE ThisFile
27     FOR LineNum ← 1 TO 3
28        OUTPUT Buffer[LineNum]
29     NEXT LineNum
30  ENDPROCEDURE
```

(a) There is an error in the algorithm. In certain cases, a text file will have at least three lines but the output will be incorrect.

(i) State how the output may be incorrect.

.....

..... [1]

(ii) Describe the error in the algorithm **and** explain how it may be corrected.

Description .....

.....

.....

.....

Explanation .....

.....

.....

.....

[4]

(b) The original algorithm is implemented and sometimes the last three lines of the text file are output correctly.

State the condition that results in the correct output.

.....

..... [1]

(c) Lines 20 to 23 inclusive could be replaced with a single pseudocode statement.

Write the pseudocode statement.

.....

..... [2]

8 The following diagram shows the incomplete waterfall model of the program development life cycle.

(a) Complete the diagram.

[3]

(b) Explain the meaning of the downward and upward arrows.

Downward arrows .....

.....

.....

Upward arrows .....

.....

.....

[2]

(c) Identify another type of model for the program development life cycle.

..... [1]

BLANK PAGE

9 A program allows a user to save passwords used to log in to websites. A stored password is then inserted automatically when the user logs in to the corresponding website.

A student is developing a program to generate a strong password. The password will be of a fixed format, consisting of **three groups of four** alphanumeric characters, separated by the hyphen character '-'.

An example of a password is: "FxAf-3hzV-Aq49"

A valid password:

- must be 14 characters long
- must be organised as three groups of four alphanumeric characters. The groups are separated by hyphen characters
- may include duplicated characters, **provided** these appear in different groups.

The programmer has started to define program modules as follows:

Module	Description
`RandomChar()`	• Generates a single random character from within one of the following ranges: ○ 'a' to 'z' ○ 'A' to 'Z' ○ '0' to '9' • Returns the character
`Exists()`	• Takes two parameters: ○ a string ○ a character • Performs a case-sensitive search for the character in the string • Returns `TRUE` if the character occurs in the string, otherwise returns `FALSE`
`Generate()`	• Generates a valid password • Uses `RandomChar()` and `Exists()` • Returns the password

Note: in a case-sensitive comparison, 'a' is not the same as 'A'.

(a) Write pseudocode for the module `Generate()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) A global 2D array `Secret` of type `STRING` stores the passwords together with the website domain name where they are used. `Secret` contains 1000 elements organised as 500 rows by 2 columns.

Unused elements contain the empty string (`""`). These may occur anywhere in the array.

An example of part of the array is:

Array element	Value
`Secret[27, 1]`	`"www.thiswebsite.com"`
`Secret[27, 2]`	`"●●●●●●●●●●●●"`
`Secret[28, 1]`	`"www.thatwebsite.com"`
`Secret[28, 2]`	`"●●●●●●●●●●●●"`

Note:

- For security, the passwords are stored in an encrypted form, shown as `"●●●●●●●●●●●●"` in the example.
- The passwords cannot be used without being decrypted.
- You may assume that the encrypted form of a password will **not** be an empty string.

Additional modules are defined as follows:

Module	Description
`Encrypt()`	• Takes a password as a string • Returns the encrypted form of the password as a string
`Decrypt()`	• Takes an encrypted password as a string • Returns the decrypted form of the password as a string
`FindPassword()`	• Takes a website domain name as a string • Searches for the website domain name in the array `Secret` • If the website domain name is found, the decrypted password is returned • If the website domain name is not found, an empty string is returned
`AddPassword()`	• Takes two parameters as strings: a website domain name and a password • Searches for the website domain name in the array `Secret` and if not found, adds the website domain name and the encrypted password to the array • Returns `TRUE` if the website domain name and encrypted password are added to the array, otherwise returns `FALSE`

The first three modules have been written.

Write pseudocode for the module `AddPassword()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(c) The content of the array `Secret` is to be stored in a text file for backup.

It **must** be possible to read the data back from the file and extract the website domain name and the encrypted password.

Both the website domain name and encrypted password are stored in the array as strings of characters.

The encrypted password may contain any character from the character set used and the length of both the encrypted password and the website domain name is variable.

Explain how a single line of the text file can be used to store the website domain name and the encrypted password.

.....

.....

.....

.....

.....

..... [3]

BLANK PAGE

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

7848782 5O

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2017**

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 **14** printed pages and **2** blank pages.

DC (NH) 133470/3
© UCLES 2017

CAMBRIDGE International Examinations

1 **(a)** **(i)** Procedural high-level languages usually support different data types.

Give an appropriate data type for each data value in the following table.

Data value	Data type
27	
"27"	
"27.3"	
TRUE	
27/3/2015	
27.3	

[6]

(ii) State an appropriate data structure to store the individual test scores for a class of students.

.....[1]

(iii) Describe how characters are represented using the ASCII character set.

.....

.....

.....

.....[2]

(b) Functions and procedures are subroutines.

Explain why you should use subroutines when designing a program solution.

.....

.....

.....

.....[2]

(c) The following pseudocode is an example of nested `IF` statements.

```
IF MyVar = 1
   THEN
      CALL Proc1()
   ELSE
      IF MyVar = 2
         THEN
            CALL Proc2()
         ELSE
            IF MyVar = 3
               THEN
                  CALL Proc3()
               ELSE
                  OUTPUT "Error"
            ENDIF
      ENDIF
ENDIF
```

Use **pseudocode** to write a `CASE` statement with the same functionality.

.....

.....

.....

.....

.....

.....

.....

.....[4]

(d) Program coding is a transferable skill.

You are given program code written in a high-level language that you have not studied.

State **two** different features of the code that you should be able to recognise.

1 .....

.....

2 .....

.....

[2]

2 The following pseudocode represents a simple algorithm.

```
DECLARE NumberFound, Remainder, Number : INTEGER
DECLARE StartNumber, EndNumber, Divisor : INTEGER

INPUT StartNumber
INPUT EndNumber
INPUT Divisor
NumberFound ← 0

FOR Number ← StartNumber TO EndNumber
   Remainder ← MODULUS(Number, Divisor)
   IF Remainder = 0
      THEN
         OUTPUT Number
         NumberFound ← NumberFound + 1
   ENDIF
ENDFOR
OUTPUT "Count: " & NumberFound
```

For the built-in functions list, refer to the **Appendix** on page 14.

(a) Complete the following trace table.

StartNumber	EndNumber	Divisor	NumberFound	Number	Remainder	Output
11	13	2	0			

[3]

(b) Describe the purpose of this algorithm.

.....

.....

.....

.....

.....

.....[3]

(c) Draw a program flowchart to represent this algorithm.

Variable declarations are **not** required in program flowcharts.

[10]

3 **(a)** A multi-user computer system stores information about users. It uses a 1D array, `UserNameArray`, of type `STRING`. There are 100 elements in the array.

The format of the string in each element of the array is as follows:

```
<UserID><UserName>
```

- `UserID` is a six-character string of numerals.
- `UserName` is a variable-length string.

Write **pseudocode** for a procedure, `BubbleSort`, to perform an efficient bubble sort on `UserNameArray`. The array is to be sorted in ascending order of `UserID`.

You should assume that `UserNameArray` has been declared as a global variable.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

(b) The value of `UserID` should be unique for each user but a problem has occurred and repeated `UserID` values may have been issued.

The array is sorted by `UserID`, so any repeated `UserID` values will appear in consecutive array elements.

A procedure, `FindRepeats` is required.

This will:

- compare each element with the previous element and output the `UserID` and `UserName` if the `UserID` is repeated
- output the total number of `UserID`s that are repeated.

For example, the `UserNameArray` contains the following entries.

Array element	Comment
⋮	
`122222Jim Moriarty`	
⋮	
`123456Fred Smith`	
`123456Eric Sykes`	Repeated User ID
`123456Kevin Turvey`	Repeated User ID
⋮	
`222244Alice Chan`	
`222244Myra Singh`	Repeated User ID
⋮	
`333333Yasmin Halim`	
⋮	

For this example, the output is:

```
123456Eric Sykes
123456Kevin Turvey
222244Myra Singh
There are 3 repeated UserIDs
```

If no repeated `UserID`s are found, the output is:

```
The array contains no repeated UserIDs
```

Write **program code** for the procedure, `FindRepeats`.
You should assume that `UserNameArray` has been declared as a global variable.

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]

(c) **(i)** The `FindRepeats` procedure forms part of a program.

Name **three** stages in a program development cycle.

1 .....

2 .....

3 .....
[3]

(ii) The program containing `FindRepeats` will be created using an IDE.

State what is meant by IDE.

.....

.....[1]

(iii) Name **two** features provided by an IDE that assist in the program development cycle.

1 .....

.....

2 .....

.....
[2]

(iv) The procedure, `FindRepeats`, is written assuming there are 100 elements in `UserNameArray`.

In the main program, the global array, `UserNameArray`, has been declared with only 50 elements.

State the type of error this will cause.

.....[1]

4 Numeric formatting converts a numeric value to a string in order to present it in a specific way.

In a generic high-level language, formatting is implemented using a mask system. In this system, each character of the mask corresponds to one character of the formatted string.

Mask characters have the following meaning:

Mask character	Meaning
#	Character must be a digit or a space
0	Character must be a digit

Any other mask characters are taken as literal values and are included in the formatted string.

(a) Using the mask `"###00.00"`, complete the following table. Use □ to represent a space. The first value has been done for you.

Value	Formatted string
1327.5	"□1327.50"
1234	
7.456	

[2]

(b) For each row in the following table, define the mask required to produce the formatted output from the given value. □ represents a space.

Value	Required output	Mask
1234.00	"1,234.00"	
3445.66	"£3,445.66"	
10345.56	"$□□10,345"	

[3]

5 A sports club maintains a record of the email address of each of its members. The details are stored in a text file, `EmailDetails.txt`. The format of each line of the text file is as follows:

```
<MembershipNumber><EmailAddress>
```

- `MembershipNumber` is a four-character string of numerals.
- `EmailAddress` is a variable-length string.

Membership of the club has increased and a four-character membership number is no longer adequate.

A procedure, `MakeNewFile`, is required to perform the following actions:

1. Create a new file, `NewEmailDetails.txt`
2. Read a line from file `EmailDetails.txt`
3. Extend `MembershipNumber` by adding two leading zero digits (for example, `"1234"` becomes `"001234"`)
4. Write the new line to file `NewEmailDetails.txt`
5. Repeat steps 2 to 4 for all lines in the original file.

(a) Write **pseudocode** for the procedure `MakeNewFile`.

For the built-in functions list, refer to the **Appendix** on page 14.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

(b) A function, `IsEmailValid`, is to be written to test for a valid email address format.

An email address has a valid format if it obeys the following three rules:

1. It contains a single '@' symbol.
2. The '@' symbol must be preceded by at least one character.
3. The '@' symbol must be followed by at least three characters.

Choose **three** different invalid strings to test distinct aspects of the rules.

Explain your choice in each case.

1 .....

Explanation .....

.....

.....

2 .....

Explanation .....

.....

.....

3 .....

Explanation .....

.....

.....

[6]

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

```
MODULUS(x : INTEGER, y : INTEGER) RETURNS INTEGER
```

returns the remainder when `x` is divided by `y` using integer arithmetic.
Example: `MODULUS(5, 2)` returns `1`

```
INT(x : REAL) RETURNS INTEGER
```

returns the integer part of `x`.
Example: `INT(27.5415)` returns `27`

```
LENGTH(ThisString : STRING) RETURNS INTEGER
```

returns the integer value representing the length of string `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 string `"ABC"`

```
RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING
```

returns rightmost `x` characters from `ThisString`.
Example: `RIGHT("ABCDEFGH", 3)` returns string `"FGH"`

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`

© UCLES 2017

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2017

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 4 3 2 4 5 7 1 9 5 7 *

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2017**

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 **13** printed pages and **3** blank pages.

DC (NF) 129956/4
© UCLES 2017

CAMBRIDGE International Examinations

1 **(a)** Simple algorithms usually consist of input, process and output.

The statements in the following table are in a generic programming language.

Complete the table by placing ticks in the relevant boxes.

Item	Statement	Input	Process	Output
1	`String1 = "Hello World"`			
2	`DISPLAY RIGHT(String1, 5)`			
3	`READFILE (MyFile, String2)`			
4	`WRITEFILE (MyFile, "Data is " & String2)`			

[6]

(b) (i) Complete the following two sentences.

A suitable operand type for an arithmetic operator is .....

A suitable operand type for a logical operator is .....

[2]

(ii) The following table shows the values of three variables.

Variable	Value
`FlagA`	`TRUE`
`FlagB`	`FALSE`
`FlagC`	`TRUE`

Evaluate these expressions.

Expression	Evaluates to
`(FlagA AND FlagB) OR FlagC`	
`FlagA AND (FlagB OR FlagC)`	
`(NOT FlagA) OR (NOT FlagC)`	

[3]

(c) The loop construct (also known as repetition or iteration) appears in many algorithms.

Use **pseudocode** to write a post-condition loop to output all the odd numbers between 100 and 200.

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

2 A multi-user computer system maintains a text file containing the ID and preferred name for each user.
User IDs are unique. Preferred names may be repeated.

(a) Stepwise refinement is to be applied to the following three steps.

After a user logs in, a welcome message is produced as follows:

1. Search for the user ID in the file.
2. Read the preferred name from the file.
3. Output the welcome message.

Describe the goal of **stepwise refinement**.

.....

.....

.....

..... [2]

(b) An initial identifier table is created as part of the stepwise refinement. A section of the table is shown. Complete this table.

Identifier	Data type	Description
`SearchUserID`		Stores the user ID entered
`FileUserID`		....
`FilePreferredName`		....
`IDFoundFlag`		....

[5]

(c) The file may not contain an entry for every user. If the user ID is not found, then the name "Guest" should be used as the preferred name.

Use **structured English** to write a more detailed algorithm for the welcome message process described in **part (a)**.

Use the identifiers given in the previous table.

Assume that the file, `UserNames.txt`, is organised as a simple text file. The user ID and preferred name are held together in unencrypted form, one entry per line.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

3 A string conversion function, `ExCamel`, needs to be written.

This function forms a return string, `OutString`, from a given string, `InString`, by:

1 separating the original words (a word is assumed to start with a capital letter)
2 converting all characters to lower case.

The following shows a pair of example values for the string values `InString` and `OutString`.

```
InString : "MyUserInput"
OutString : "my user input"
```

You may assume that `InString` always starts with a capital letter.

The following is a first attempt at writing the pseudocode for this function.

Complete the **pseudocode** using appropriate built-in functions.

For the built-in functions list, refer to the **Appendix** on page 13.

```
FUNCTION ExCamel (............................) RETURNS .............

   DECLARE NextChar : ........................................

   DECLARE ...................................................... : STRING

   DECLARE n: INTEGER

   ........................................ // initialise the return string

   // loop through InString to produce OutString

   FOR n ← 1 TO ............................. // from first to last

      NextChar ← ............................ // get next character

      IF .................................... // check if upper case

         THEN

            IF n > 1                          // if not first character

               THEN

                  ............................ // add space to OutString

            ENDIF

            ................................... // make NextChar lower case

      ENDIF

      ...................................... // add NextChar to OutString

   ENDFOR

   ........................................... // return value

ENDFUNCTION
```

[11]

4 **(a)** High-level programming languages have many features that support the modular approach. One such feature is the use of parameters.

State **two** other features.

1 .....

.....

2 .....

.....

[2]

(b) Consider the following pseudocode.

```
PROCEDURE MyProc(x)
   x ← x + 1
ENDPROCEDURE

Intermediate lines of pseudocode not shown

x ← 4
CALL MyProc(x)
OUTPUT(x)
```

Parameter `x` is used to pass data to procedure `MyProc`.
There are two parameter passing methods that could be used.

Complete the following table for each of the two methods.

Name of parameter passing method	Value output	Explanation
....	.....	....
....	.....	....

[6]

5 A multi-user computer system records user login data. Each time a user successfully logs into the system, it records the following data.

Data item	Example data
User ID	`"Jim27"`
Port ID	`"3456"`
Time and date	`"08:30 Jun 01 2015"`

The data items are concatenated (joined) using a separator character to form a single string. Each string represents one log entry.

(a) **(i)** Suggest a suitable separator character. Give the reason for your choice.

Character .....

Reason .....

.....
[2]

(ii) The concatenated strings are stored in an array, `LogArray`, which may contain up to 20 log entries.

Use **pseudocode** to declare `LogArray`.

..... [2]

(b) A procedure, `LogEvents`, is required to add the log entry data from `LogArray` to the existing text file, `LoginFile.txt`.

Unused array elements are assigned the value `"****"`. These can occur anywhere in the array and should not be written to the file.

Write **program code** for the procedure `LogEvents`.

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]

6 A computerised vehicle licensing system stores details about vehicles and their registration marks (number plates or license plates).

To be valid, a vehicle registration must comply with the following rules:

- It must be between six and nine characters long.
- Characters 1 to 3 are upper case alphabetic characters.
- Characters 4 to 5 are numeric characters.
- Remaining characters are upper case alphabetic.

A function, `ValidateRegistration` is needed to check that a given registration mark follows these rules. This function takes a string, `Registration` as a parameter and returns a Boolean value:

- `TRUE` if it is a valid registration
- `FALSE` otherwise.

(a) Write **program code** to implement the new function, `ValidateRegistration`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [9]

(b) The function is to be tested.

Give a valid string that could be used to check that the function returns TRUE under the correct conditions.

String1: .....

Modify your valid String1 to test each rule separately.

Explain your choice in each case.

String2: .....

Explanation: .....

.....

.....

String3: .....

Explanation: .....

.....

.....

String4: .....

Explanation: .....

.....

.....

String5: .....

Explanation: .....

.....

.....

[5]

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

```
RIGHT(ThisString : STRING, x : INTEGER) RETURNS STRING
```

returns rightmost `x` characters from `ThisString`.
Example: **`RIGHT("ABCDEFGH", 3)`** returns string **`"FGH"`**

```
LENGTH(ThisString : STRING) RETURNS INTEGER
```

returns the integer value representing the length of string `ThisString`.
Example: **`LENGTH("Happy Days")`** returns **`10`**

```
MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING
```

returns string of length `y` starting at position `x` from `ThisString`.
Example: **`MID("ABCDEFGH", 2, 3)`** returns string **`"BCD"`**

```
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 then it is returned unchanged.
Example: **`LCASE('W')`** 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 then it is returned unchanged.
Example: **`UCASE('h')`** returns **`'H'`**

```
MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```

returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`.
Example: **`MOD(10,3)`** returns **`1`**

```
DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```

returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`.
Example: **`DIV(10,3)`** returns **`3`**

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings Example: **`"Summer" & " " & "Pudding"`** produces **`"Summer Pudding"`**
`AND`	Performs a logical **`AND`** of two Boolean values Example: **`TRUE AND FALSE`** produces **`FALSE`**
`OR`	Performs a logical **`OR`** of two Boolean values Example: **`TRUE OR FALSE`** produces **`TRUE`**

© UCLES 2017

BLANK PAGE

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2017

Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 9 7 4 4 6 8 1 1 6 1 *

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.

© UCLES 2019

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]

(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'`

© UCLES 2019

`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`

© UCLES 2019

BLANK PAGE

BLANK PAGE

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.

© UCLES 2019

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2023**

INSERT **2 hours**

1790695172–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (ST) 333153
© UCLES 2023

An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and character functions

- A string of length 1 may be considered to be either of type `CHAR` or `STRING`
- A `CHAR` may be assigned to, or concatenated with, a `STRING`
- A `STRING` of length greater than 1 cannot be assigned to a `CHAR`

```
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"`

```
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`

```
TO_UPPER(x : <datatype>) RETURNS <datatype>
```

`<datatype>` may be `CHAR` or `STRING`

returns an object of type `<datatype>` formed by converting all characters of `x` to upper case.

Examples:

- `TO_UPPER("Error 803")` returns `"ERROR 803"`
- `TO_UPPER('a')` returns `'A'`

```
TO_LOWER(x : <datatype>) RETURNS <datatype>
```

`<datatype>` may be `CHAR` or `STRING`

returns an object of type `<datatype>` formed by converting all characters of `x` to lower case.

Examples:

- `TO_LOWER("JIM 803")` returns `"jim 803"`
- `TO_LOWER('W')` returns `'w'`

```
NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>
```

returns a string representation of a numeric value.
`<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING`
Example: `NUM_TO_STR(87.5)` returns `"87.5"`
If `x` is a negative value, the returned value will be a string beginning with the '–' character.

```
STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>
```

returns a numeric representation of a string.
`<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER`
Example: `STR_TO_NUM("23.45")` returns `23.45`

```
IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN
```

returns `TRUE` if `ThisString` represents a valid numeric value.
`<datatype>` may be `CHAR` or `STRING`
Example: `IS_NUM("-12.36")` returns `TRUE`

```
ASC(ThisChar : CHAR) RETURNS INTEGER
```

returns an integer value (the ASCII value) of `ThisChar`
Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.

```
CHR(x : INTEGER) RETURNS CHAR
```

returns the character whose integer value (the ASCII value) is `x`
Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric functions

```
INT(x : REAL) RETURNS INTEGER
```

returns the integer part of `x`
Example: `INT(27.5415)` returns `27`

```
RAND(x : INTEGER) RETURNS REAL
```

returns a real number in the range `0` to `x` (**not** inclusive of `x`).
Example: `RAND(87)` could return `35.430729`

Date functions
Date format is assumed to be DD/MM/YYYY unless otherwise stated

```
DAY(ThisDate : DATE) RETURNS INTEGER
```

returns the current day number from `ThisDate`
Example: `DAY(04/10/2003)` returns `4`

```
MONTH(ThisDate : DATE) RETURNS INTEGER
```

returns the current month number from `ThisDate`
Example: `MONTH(04/10/2003)` returns `10`

```
YEAR(ThisDate : DATE) RETURNS INTEGER
```

returns the current year number from `ThisDate`
Example: `YEAR(04/10/2003)` returns `2003`

```
DAYINDEX(ThisDate : DATE) RETURNS INTEGER
```

returns the day index number from `ThisDate` where Sunday = 1, Monday = 2 etc.
Example: `DAYINDEX(07/11/2023)` returns 3

```
SETDATE(Day, Month, Year : INTEGER) RETURNS DATE
```

returns a value of type `DATE` with the value of `<Day>/<Month>/<Year>`
Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26/10/2003

```
TODAY() RETURNS DATE
```

returns a value of type `DATE` corresponding to the current date.

Text file functions

```
EOF(FileName : STRING) RETURNS BOOLEAN
```

returns `TRUE` if there are no more lines to be read from file `FileName`
The function will generate an error if the file is not already open in `READ` mode.

Operators

An error will be generated if an operator is used with a value(s) of an incorrect type.

`&`	concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values. Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values. Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value. Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another. Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another. Example `10 DIV 3` evaluates to `3`

Comparison operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	used to compare two items of the same type.
	evaluates to `TRUE` if the condition is true, otherwise evaluates to `FALSE`
	
	Notes:
	
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` and / or `STRING`
	• cannot be used to compare two records
	
	Examples:
	
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when `Count` contains the value 4

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 2 5 4 3 3 5 7 7 0 8 *

COMPUTER SCIENCE **9608/22**

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 **16** printed pages.

DC (ST/CT) 163546/4
© UCLES 2019

Cambridge Assessment International Education

[Turn over

1 **(a)** Algorithms are used in computer programming.

(i) Explain the term **algorithm**.

.....

.....

.....

..... [2]

(ii) Algorithms usually consist of three different types of activity.

Complete the table below.

The third activity type is given.

Activity type	Pseudocode example
.....	....
.....	....
Output	....

[5]

(b) Program variables have values as follows:

Variable	Value
Married	03/04/1982
ID	"M1234"
MiddleInitial	'J'
Height	5.6
IsMarried	TRUE
Children	2

(i) 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 16.

Expression	Evaluates to
STRING_TO_NUM(RIGHT(ID, 3))	
INT(Height * Children)	
IsMarried AND Married < 31/12/1999	
LENGTH(ID & NUM_TO_STRING(Height))	
MID(ID, INT(Height) - Children, 2)	

[5]

(ii) Programming languages support different data types.

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

Variable	Data type
Married	
ID	
MiddleInitial	
Height	
IsMarried	

[5]

2 **(a)** **(i)** Procedures and functions are examples of subroutines.

State a reason for using subroutines in the construction of an algorithm.

.....

..... [1]

(ii) Give **three** advantages of using subroutines in a program.

1 .....

.....

2 .....

.....

3 .....

.....
[3]

(iii) The following pseudocode uses the subroutine `DoSomething()`.

```
Answer ← 23 + DoSomething("Yellow")
```

State whether the subroutine is a function or a procedure. Justify your answer.

Type of subroutine .....

Justification .....

.....
[2]

(b) The program development cycle involves writing, translating and testing a high-level language program.

Describe these activities with reference to **each** of the following:

- editor
- translator
- debugger

.....

.....

.....

.....

.....

..... [3]

(c) The following lines of code are taken from a high-level language program.

```
WHEN Result < 20
  {
   Call ResetSensor(3)
   Result := GetSensor(3)
  }
```

Identify the type of control structure **and** describe the function of the code.

Control structure .....

Function of code .....

.....

.....

.....

.....

.....
[3]

3 The following structure chart shows the relationship between three modules.

Parameters `A` to `E` have the following data types:

```
A, D : STRING
C    : CHAR
B, E : INTEGER
```

(a) **(i)** Write the **pseudocode** header for module `SubA()`.

.....

.....

..... [3]

(ii) Write the **pseudocode** header for module `SubB()`.

.....

.....

..... [3]

(b) Module hierarchy and parameters are two features that may be represented on a structure chart.

State **two other** features than can be represented.

Feature 1 .....

Feature 2 .....
[2]

Question 4 begins on the next page.

4 The following is pseudocode for a string handling function.

For the built-in functions list, refer to the **Appendix** on page 16.

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

   DECLARE NewString : STRING
   DECLARE Index : INTEGER
   DECLARE NextChar : CHAR
   DECLARE Selected : INTEGER
   DECLARE NewValue : INTEGER

   NewString ← '0'
   Selected ← 0

   FOR Index ← 1 TO LENGTH(InString)

      NextChar ← MID(InString, Index, 1)
      IF NextChar < '0' OR NextChar > '9'
         THEN
            NewValue ← STRING_TO_NUM(NewString)
            IF NewValue > Selected
               THEN
                  Selected ← NewValue
            ENDIF
            NewString ← '0'
         ELSE
            NewString ← NewString & NextChar
      ENDIF

   ENDFOR

   RETURN Selected

ENDFUNCTION
```

(a) **(i)** The following assignment calls the `Search()` function:

```
Result ← Search("12∇34∇5∇∇39")
```

Complete the following trace table by performing a dry run of this function call.

The symbol `'∇'` represents a space character. Use this symbol to represent a space character in the trace table.

Index	NextChar	Selected	NewValue	NewString

[5]

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

..... [1]

(b) There is an error in the algorithm. When called as shown in **part (a)(i)**, the function did not return the largest value as expected.

(i) Explain why this error occurred when the program called the function.

.....

.....

.....

..... [2]

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

.....

.....

.....

..... [2]

5 A student is learning about text files. She wants to write a program to count the number of lines in a file.

(a) Use **structured English** to describe an algorithm she could use.

.....

.....

.....

.....

.....

..... [3]

(b) A procedure, `CountLines()`, is being written to count the number of lines in a text file. The procedure will:

- take a filename as a string parameter
- count the number of lines in the file
- output a single suitable message that includes the total number of lines.

Write **pseudocode** for the procedure `CountLines()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

6 Nadine is developing a program to store the ID and preferred name for each student in a school. For example, student Pradeep uses the preferred name “Prad”.

The program will:

1. prompt and input a valid user ID and a preferred name
2. write the user ID and preferred name to one of two files
3. allow the user to end the program or repeat from step 1.

The program will consist of three separate modules. Each module will be implemented using either a procedure or a function.

Nadine has defined the modules as follows:

Module	Description
`TopLevel()`	• Call `GetInfo()` to obtain a string containing a valid user ID and a preferred name • Call `WriteInfo()` to write the string to either `File1.txt` or `File2.txt` depending on the first character of the user ID as follows: ○ ‘A’ to ‘M’: writes to `File1.txt` ○ ‘N’ to ‘Z’: writes to `File2.txt` For example, a string with a user ID of `"G1234"` writes to `File1.txt` • End the program if the file write was unsuccessful • Input (Y/N) to either repeat for the next user ID or to end the program
`GetInfo()`	• Input a user ID and repeat until the user ID is valid • Input a preferred name. This will be an empty string if no preferred name is input. • Concatenate the user ID and preferred name using a `'*'` character as a separator and return this string
`WriteInfo()`	• Open the file • Append the concatenated string to the file • Close the file • Return a Boolean value: ○ `TRUE` if the file write was successful ○ `FALSE` if the file write failed, for example, if the disk was full

A valid user ID:

- is five characters in length
- has a single upper case alphabetic character followed by four numeric characters, for example “G1234”.

Nadine has decided that global variables and nested modules must not be used.

Nadine wants all inputs to have suitable prompts.

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

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 **program code** for the module `TopLevel()`.

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]

(c) Write **pseudocode** for the module declaration of `WriteInfo()`.

.....

.....

.....

..... [3]

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.

© UCLES 2019

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(87.5)` returns `"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("23.45")` returns `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`

© UCLES 2019

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

0015260821

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.

DC (RW/FC) 205023/2
© UCLES 2021

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]

(b) The character count values have been assigned to the array in **part (a)**.

Use **structured English** to describe an algorithm to:

- search the array
- output the alphabetic character that occurs most often (the highest character count value)
- output a suitable message if more than one alphabetic character has the same highest count value.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

BLANK PAGE

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`	• repeatedly calls `Module_Y` then `Module_Z` • passes a parameter of type `REAL` to `Module_Y` • passes two parameters of type `INTEGER` to `Module_Z`
`Module_Y`	calls either `Module_YA` or `Module_YB`
`Module_Z`	called with two parameters of type `INTEGER`
`Module_YA`	• called with a parameter of type `REAL` • parameter is passed by reference
`Module_YB`	• called with a parameter of type `INTEGER` • returns a `BOOLEAN` value

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.

(a) Write **program code** to implement this **amended** procedure `ScanArray()`.

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]

(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]

BLANK PAGE

5 Alice is developing an algorithm for a game. The game will count the number of attempts by a player to guess the secret number generated by the built-in random number function.

She will use pseudocode to develop a procedure, `GuessNum()`, that will:

1. generate a secret number that is a random integer in the range 1 to 20 inclusive
2. prompt and input the player's guess
3. output a message if the guess is not the same as the secret number and repeat from step 2
4. count the number of attempts it takes to guess the secret number
5. output a message when the secret number has been guessed correctly, giving the number of attempts.

(a) **(i)** Write **pseudocode** for the procedure `GuessNum()`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(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 10 000 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
`GetValidFilename()`	• prompts, inputs and returns a valid filename
`Check()`	• called with an array index as a parameter • checks that the data values for the stock item with the given index are valid • returns `TRUE` if the values are valid, otherwise returns `FALSE`
`Backup()`	• calls `GetValidFilename()` to get the name of the backup file • combines the data values for each stock item to form a single string. Inserts an asterisk character '*' as a separator between data values • writes the string to the backup file • calls `Check()` to validate the data values. If there is an error then also writes the string to the file `"ERRORLOG.TXT"` • repeats for all stock items • returns `TRUE` if nothing was written to `"ERRORLOG.TXT"`, otherwise returns `FALSE`

`Unpack()`	• called with two parameters:
	  ○ an array index
	  ○ a string value read from one line of the backup file
	• extracts the four data values from the string and assigns each to the appropriate array

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

Data values relating to an individual stock item are valid if:

- the `StockID` element is of length 8
- the `Description` is not an empty string
- the `Quantity` is zero or above.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

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

.....

.....

.....

.....

.....

..... [8]

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

Module	Description

`Unpack()`	• called with two parameters:
	  ○ an array index
	  ○ a string value read from one line of the backup file
	• extracts the four data values from the string and assigns each to the appropriate array

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

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"`

© UCLES 2021

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`

© UCLES 2021

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

1281968342

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2023**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (EF/SG) 338525/4
© UCLES 2023

Refer to the **insert** for the list of pseudocode functions and operators.

1 A programmer has written a program which includes the function `Calculate()`.
When the program is run, the function returns an unexpected value.

(a) Describe how a typical Integrated Development Environment (IDE) could be used to help debug the program to find the errors in the function `Calculate()`.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) The algorithm for function `Calculate()` contains the three pseudocode statements shown.

Describe the error in each statement or write 'no error' if the statement contains no error.

Assume any variables used are of the correct type for the given function.

Statement 1: `Index ← STR_TO_NUM(("27") + 2)`

Error .....

.....

Statement 2: `Index ← STR_TO_NUM(MID("CPE1704TKS", 4, 2))`

Error .....

.....

Statement 3: `IF MONTH(ThisDate) > '6' THEN`

Error .....

.....
[3]

(c) The program contains variables with values as follows:

Variable	Value
`Active`	`TRUE`
`Points`	`75`
`Exempt`	`FALSE`

(i) Complete the table by evaluating each expression.

	Expression	Evaluation
1	`(Points > 99) OR Active`	
2	`(Points MOD 2 = 0) OR Exempt`	
3	`(Points <= 75) AND (Active OR Exempt)`	
4	`(Active OR NOT Active) AND NOT Exempt`	

[2]

(ii) Write expression **4** from the table in part **(c)(i)** in its simplest form.

..... [1]

2 A program contains an algorithm to output a string of a specified length containing identical characters.

(a) The algorithm is described as follows:

1. prompt and input a character and store in `MyChar`
2. prompt and input an integer and store in `MyCount`
3. generate a string consisting of `MyChar` repeated `MyCount` times
4. output the string.

Draw a program flowchart to represent the algorithm.

[4]

(b) A different part of the program uses the variable `StartDate`.

Write pseudocode statements to declare `StartDate` **and** assign to it the date corresponding to 15/11/2005.

Declaration .....

Assignment .....
[3]

3 Customers collect points every time they make a purchase at a store.

A program is used to manage the points system and the table lists some of the information stored for one customer.

Information	Data type required
Name	String
Number of points collected	Integer
Date of birth	Date

(a) **(i)** Identify a suitable structure for storing the information for one customer. Explain the advantage of using this structure.

Structure .....

Advantage .....

.....

.....

.....

.....

.....
[4]

(ii) Describe a data structure that could be used to store the information for **all** customers.

.....

..... [2]

(b) Customers receive points depending on the amount they spend. The number of points depends on the band that the amount falls into:

Band	Amount	Points
1	Less than $10	5 per whole dollar ($)
2	Between $10 and $100 inclusive	7 per whole dollar ($)
3	Over $100	10 per whole dollar ($)

For example, if the amount is $99.77, this amount is in band 2 and therefore the number of points is 7 × 99, which is 693 points.

The algorithm to calculate the points from a given amount is expressed as follows:

- work out the appropriate band
- calculate and output the number of points.

Apply the process of stepwise refinement to increase the detail of the algorithm. Structure your algorithm into a sequence of five steps that could be used to produce pseudocode.

Write the **five** steps.

1 .....

.....

.....

2 .....

.....

.....

3 .....

.....

.....

4 .....

.....

.....

5 .....

.....

.....

[5]

4 Function `Replace()` will:

1. take three parameters:
   - a string (the original string)
   - a char (the original character)
   - a char (the new character)
2. form a new string from the original string where all instances of the original character are replaced by the new character
3. return the new string.

Write pseudocode for function `Replace()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

5 Several companies are developing websites to market a new type of games console. The company that is first to create a website that can demonstrate the interactive features of the games console will have an advantage over the others. The requirements for the website are likely to change as more information about the features of the console are made available.

One company has decided to develop their website using a program development life cycle based on the waterfall model.

(a) **(i)** Give **two** reasons why this may **not** be the most appropriate model to use in this case.

Reason 1 .....

.....

Reason 2 .....

.....
[2]

(ii) Identify a **more appropriate** program development life cycle model for this scenario.

..... [1]

(b) The website has been running in test mode for several weeks.

Identify **and** describe a final stage of testing that should take place before the website is made available to all customers.

Stage .....

Description .....

.....

.....

.....

.....
[3]

6 A video-conferencing program supports up to six users. Speech from each user is sampled and digitised (converted from analogue to digital). Digitised values are stored in array `Sample`.

The array `Sample` consists of 6 rows by 128 columns and is of type integer. Each row contains 128 digitised sound samples from one user.

The digitised sound samples from each user are to be processed to produce a single value which will be stored in a 1D array `Result` of type integer. This process will be implemented by procedure `Mix()`.

A procedure `Mix()` will:

- calculate the average of each of the 6 sound samples in a column
- ignore sound sample values of 10 or less
- store the average value in the corresponding position in `Result`
- repeat for each column in array `Sample`

The diagram uses example values to illustrate the process:

Write pseudocode for procedure `Mix()`.

Assume `Sample` and `Result` are global.

..... [6]

7 A school has a computerised library system that allows students to borrow books for a fixed length of time. The system uses text files to store details of students, books and loans.

A new module is to be written which will generate emails to each student who has an overdue book.

(a) Decomposition will be used to break down the problem of designing the new module into sub-problems.

Identify **three** program modules that could be used in the design **and** describe their use.

Module 1 .....

Use .....

.....

.....

.....

Module 2 .....

Use .....

.....

.....

.....

Module 3 .....

Use .....

.....

.....

.....

[3]

(b) The program designer produces a structure chart for the new module. Part of the structure chart is shown:

(i) Explain the relationship between the four modules shown.

.....

.....

.....

..... [2]

(ii) Two new modules are added: `Module-X()` and `Module-Y()`.

- `Module-X()` has no parameters.
- `Module-Y()` will take a string and a real number as parameters and return a Boolean value.
- `Module-D()` will call either `Module-X()` or `Module-Y()`.

Draw **only** the part of the structure chart that represents the relationship between `Module-X()`, `Module-Y()` and `Module-D()`.

[3]

8 A computer shop assembles computers using items bought from several suppliers. A text file `Stock.txt` contains information about each item.

Information for each item is stored as a single line in the `Stock.txt` file in the format:

```
<ItemNum><SupplierCode><Description>
```

Item information is as follows:

	Format	Comment
`ItemNum`	4 numeric characters	unique for each item in the range "0001" to "5999" inclusive
`SupplierCode`	5 alphabetic characters	to identify the supplier of the item
`Description`	a string	a minimum of 12 characters

The file is organised in ascending order of `ItemNum` and does **not** contain all possible values in the range.

A programmer has started to define program modules as follows:

Module	Description
`SuppExists()` (already written)	• called with a parameter of type string representing a supplier code • returns `TRUE` if the supplier code is already in use, otherwise returns `FALSE`
`IsNewSupp()`	• called with a parameter of type string representing a new supplier code • returns `TRUE` if the string only contains alphabetic characters (either upper or lower case) and the supplier code is **not** already in use, otherwise returns `FALSE`

(a) Write pseudocode for module `IsNewSupp()`.

Module `SuppExists()` has already been written and should be used as part of your solution. Module `SuppExists()` will generate a run-time error if the given parameter is not 5 characters in length.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) A new module has been defined:

Module	Description
`CheckNewItem()`	• called with a parameter of type string representing a line of item information • checks to see whether an item with the same `ItemNum` already exists in the file • returns `TRUE` if the `ItemNum` is not already in the file, otherwise returns `FALSE`

Write **efficient** pseudocode for module `CheckNewItem()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(c) The program modules `SuppExists()`, `IsNewSupp()` and `CheckNewItem()` are part of a group of modules that are combined to create a complete stock control program.

Each module in the program is tested individually during development and is debugged as necessary. It is then added to the program and further testing performed.

(i) Identify this method of testing.

..... [1]

(ii) One of the modules does not work properly when it is added to the program.

Describe a testing method that can be used to address this problem so that testing can continue and other modules can be added.

.....

.....

.....

..... [2]

(d) A new module `AddItem()` will be used to add information to the `Stock.txt` file.

State the file mode that should be used for the algorithm within this module.

..... [1]

(e) A new module `FindItem()` searches for a given item in the `Stock.txt` file, which is already organised in ascending order of `ItemNum`.

Describe how this organisation may improve the efficiency of the algorithm.

.....

.....

.....

.....

.....

..... [3]

BLANK PAGE

BLANK PAGE

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2022**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (PQ) 302745/3
© UCLES 2022

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** An algorithm includes a number of complex calculations. A programmer is writing a program to implement the algorithm and decides to use library routines to provide part of the solution.

State **three** possible benefits of using library routines in the development of the program.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(b) The following pseudocode is part of a program that stores names and test marks for use in other parts of the program.

```
DECLARE Name1, Name2, Name3 : STRING
DECLARE Mark1, Mark2, Mark3 : INTEGER
INPUT Name1
INPUT Mark1
INPUT Name2
INPUT Mark2
INPUT Name3
INPUT Mark3
```

(i) The pseudocode needs to be changed to allow for data to be stored for up to 30 students.

Explain why it would be good practice to use arrays to store the data.

.....

.....

.....

.....

.....

..... [3]

(ii) The following pseudocode statement includes array references:

```
OUTPUT "Student ", Name[Count], " scored ", Mark[Count]
```

State the purpose of the variable `Count` and give its data type.

Purpose .....

.....

Data type .....

[2]

(c) The pseudocode statements in the following table may contain errors.

State the error in each case or write 'NO ERROR' if the statement contains no error.

Assume that any variables used are of the correct type for the given function.

Statement	Error
`IF EMPTY ← "" THEN`	
`Status ← IS_NUM(-23.4)`	
`X ← STR_TO_NUM("37") + 5`	
`Y ← STR_TO_NUM("37" + "5")`	

[4]

2 A system is being developed to help manage a car hire business. A customer may hire a car for a number of days.

An abstract model needs to be produced.

(a) Explain the process of abstraction **and** state **four** items of data that should be stored each time a car is hired.

Explanation .....

.....

Item 1 .....

Item 2 .....

Item 3 .....

Item 4 .....
[3]

(b) Identify **two** operations that would be required to process the car hire data.

Operation 1 .....

.....

Operation 2 .....

.....
[2]

3 A 1D array `Data` of type integer contains 200 elements. Each element has a unique value.

An algorithm is required to search for the largest value and output it.

Describe the steps that the algorithm should perform.

Do **not** include pseudocode statements in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

4 **(a)** The following diagram shows an Abstract Data Type (ADT) representation of an ordered linked list. The data item stored in each node is a single character. The data will be accessed in alphabetical order.

The symbol Ø represents a null pointer.

Start pointer

(i) Nodes with data 'A' and 'K' are added to the linked list. Nodes with data 'J' and 'L' are deleted.

After the changes, the data items still need to be accessed in alphabetical order.

Complete the diagram to show the new state of the linked list.

[4]

(ii) The original data could have been stored in a 1D array in which each element stores a character.

For example:

'C'	'J'	'L'		

Explain the advantages of making the changes described in **part (a)(i)** when the data is stored in the linked list instead of an array.

.....

.....

.....

..... [2]

(iii) Explain the disadvantages of making the changes described in **part (a)(i)** when the data is stored in the linked list instead of an array.

.....

.....

.....

..... [2]

(b) A program will store data using a linked list like the one shown in **part (a)**.

Explain how the linked list can be implemented.

.....

.....

.....

.....

.....

.....

.....

..... [4]

5 A program uses two 1D arrays of type integer. `Array1` contains 600 elements and `Array2` contains 200 elements.

`Array1` contains sample values read from a sensor. The sensor always takes three consecutive samples and all of these values are stored in `Array1`.

A procedure `Summarise()` will calculate the average of three consecutive values from `Array1` and write the result to `Array2`. This will be repeated for all values in `Array1`.

The diagram below illustrates the process for the first six entries in `Array1`.

Write pseudocode for the procedure `Summarise()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

BLANK PAGE

6 The following pseudocode algorithm attempts to check whether a string is a valid email address.

```
FUNCTION IsValid(InString : STRING) RETURNS BOOLEAN
   DECLARE Index, Dots, Ats, Others : INTEGER
   DECLARE NextChar : CHAR
   DECLARE Valid : BOOLEAN

   Index ← 1
   Dots ← 0
   Ats ← 0
   Others ← 0
   Valid ← TRUE

   REPEAT
      NextChar ← MID(InString, Index, 1)
      CASE OF NextChar
         '.' : Dots ← Dots + 1
         '@' : Ats ← Ats + 1
               IF Ats > 1 THEN
                  Valid ← FALSE
               ENDIF
         OTHERWISE : Others ← Others + 1
      ENDCASE

      IF Dots > 1 AND Ats = 0 THEN
         Valid ← FALSE
      ELSE
         Index ← Index + 1
      ENDIF

   UNTIL Index > LENGTH(InString) OR Valid = FALSE

   IF NOT (Dots >= 1 AND Ats = 1 AND Others > 8) THEN
      Valid ← FALSE
   ENDIF

   RETURN Valid

ENDFUNCTION
```

(a) Part of the validation is implemented by the line:

```
IF NOT (Dots >= 1 AND Ats = 1 AND Others > 8) THEN
```

State the values that would result in the condition evaluating to `TRUE`.

.....

.....

..... [1]

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

```
Result ← IsValid("Liz.123@big@net")
```

Index	NextChar	Dots	Ats	Others	Valid

[5]

(ii) State the value returned when `IsValid()` is called using the expression shown in **part (b)(i)**.

..... [1]

7 A simple arithmetic expression is stored as a string in the format:

<Value1><Operator><Value2>

An operator character is one of the following: `'+'` `'−'` `'*'` `'/'`

Example arithmetic expression strings:

```
"803+1904"
"34/7"
```

(a) A procedure `Calculate()` will:

- take an arithmetic expression string as a parameter
- evaluate the expression
- output the result.

Assume:

- the string contains only numeric digits and a single operator character
- Value1 and Value2 represent integer values
- Value1 and Value2 are unsigned (they will not be preceded by `'+'` or `'−'`).

(i) Write pseudocode for the procedure `Calculate()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(ii) `Calculate()` is changed to a function that returns the value of the evaluated expression.

Write the header for the function in pseudocode.

.....

..... [1]

(b) A string representing an arithmetic expression could be in the correct format but be impossible to evaluate.

Give an example of a correctly formatted string and explain why evaluation would be impossible.

Example string .....

Explanation .....

.....

.....
[2]

8 A teacher is designing a program to perform simple syntax checks on programs written by students. Student programs are submitted as text files, which are known as project files.

A project file may contain blank lines.

The teacher has defined the first program module as follows:

Module	Description
`CheckFile()`	• takes the name of an existing project file as a parameter of type string • returns `TRUE` if the file is valid (it contains at least 10 non-blank lines), otherwise returns `FALSE`

(a) Write pseudocode for module `CheckFile()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

Further modules are defined as follows:

Module	Description
`CheckLine()`	• takes a line from a project file as a parameter of type string • returns zero if the line is blank or contains no syntax error, otherwise returns an error number as an integer

`CountErrors()`	• takes two parameters:
	  ○ the name of a project file as a string
	  ○ the maximum number of errors as an integer
	• uses `CheckFile()` to test the project file. Outputs an error message and ends if the project file is not valid
	• calls `CheckLine()` for each line in the project file
	• counts the number of errors
	• outputs the number of errors or a warning message if the maximum number of errors is exceeded

(b) `CountErrors()` is called to check the project file `Jim01Prog.txt` and to stop if more than 20 errors are found.

Write the pseudocode statement for this call.

.....

..... [2]

(c) Write pseudocode for module `CountErrors()`. Assume `CheckFile()` and `CheckLine()` have been written and can be used in your solution.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(d) Module `CheckLine()` includes a check for syntax errors.

Two examples of syntax error that **cannot** be detected from examining a **single** line are those involving selection and iteration.

Give **two other** examples.

1 .....

.....

.....

2 .....

.....

.....

[2]

BLANK PAGE

BLANK PAGE

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2023**

INSERT **2 hours**

4026563148–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (KS) 333152
© UCLES 2023

An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and character functions

- A string of length 1 may be considered to be either of type `CHAR` or `STRING`
- A `CHAR` may be assigned to, or concatenated with, a `STRING`
- A `STRING` of length greater than 1 cannot be assigned to a `CHAR`

```
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"`

```
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`

```
TO_UPPER(x : <datatype>) RETURNS <datatype>
```

`<datatype>` may be `CHAR` or `STRING`

returns an object of type `<datatype>` formed by converting all characters of `x` to upper case.

Examples:

- `TO_UPPER("Error 803")` returns `"ERROR 803"`
- `TO_UPPER('a')` returns `'A'`

```
TO_LOWER(x : <datatype>) RETURNS <datatype>
```

`<datatype>` may be `CHAR` or `STRING`

returns an object of type `<datatype>` formed by converting all characters of `x` to lower case.

Examples:

- `TO_LOWER("JIM 803")` returns `"jim 803"`
- `TO_LOWER('W')` returns `'w'`

```
NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>
```

returns a string representation of a numeric value.
`<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING`
Example: `NUM_TO_STR(87.5)` returns `"87.5"`
If `x` is a negative value, the returned value will be a string beginning with the '–' character.

```
STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>
```

returns a numeric representation of a string.
`<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER`
Example: `STR_TO_NUM("23.45")` returns `23.45`

```
IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN
```

returns `TRUE` if `ThisString` represents a valid numeric value.
`<datatype>` may be `CHAR` or `STRING`
Example: `IS_NUM("-12.36")` returns `TRUE`

```
ASC(ThisChar : CHAR) RETURNS INTEGER
```

returns an integer value (the ASCII value) of `ThisChar`
Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.

```
CHR(x : INTEGER) RETURNS CHAR
```

returns the character whose integer value (the ASCII value) is `x`
Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric functions

```
INT(x : REAL) RETURNS INTEGER
```

returns the integer part of `x`
Example: `INT(27.5415)` returns `27`

```
RAND(x : INTEGER) RETURNS REAL
```

returns a real number in the range `0` to `x` (**not** inclusive of `x`).
Example: `RAND(87)` could return `35.430729`

Date functions
Date format is assumed to be DD/MM/YYYY unless otherwise stated

```
DAY(ThisDate : DATE) RETURNS INTEGER
```

returns the current day number from `ThisDate`
Example: `DAY(04/10/2003)` returns `4`

```
MONTH(ThisDate : DATE) RETURNS INTEGER
```

returns the current month number from `ThisDate`
Example: `MONTH(04/10/2003)` returns `10`

```
YEAR(ThisDate : DATE) RETURNS INTEGER
```

returns the current year number from `ThisDate`
Example: `YEAR(04/10/2003)` returns `2003`

```
DAYINDEX(ThisDate : DATE) RETURNS INTEGER
```

returns the day index number from `ThisDate` where Sunday = 1, Monday = 2 etc.
Example: `DAYINDEX(07/11/2023)` returns 3

```
SETDATE(Day, Month, Year : INTEGER) RETURNS DATE
```

returns a value of type `DATE` with the value of `<Day>/<Month>/<Year>`
Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26/10/2003

```
TODAY() RETURNS DATE
```

returns a value of type `DATE` corresponding to the current date.

Text file functions

```
EOF(FileName : STRING) RETURNS BOOLEAN
```

returns `TRUE` if there are no more lines to be read from file `FileName`
The function will generate an error if the file is not already open in `READ` mode.

Operators

An error will be generated if an operator is used with a value(s) of an incorrect type.

`&`	concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values. Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values. Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value. Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another. Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another. Example `10 DIV 3` evaluates to `3`

Comparison operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	used to compare two items of the same type.
	evaluates to `TRUE` if the condition is true, otherwise evaluates to `FALSE`
	
	Notes:
	
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` and / or `STRING`
	• cannot be used to compare two records
	
	Examples:
	
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when `Count` contains the value 4

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

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 **October/November 2020**

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 **16** pages. Blank pages are indicated.

DC (JC/CT) 188579/4
© UCLES 2020

1 **(a)** A programmer uses the process of stepwise refinement to break down a problem.

Explain the purpose of stepwise refinement.

.....

.....

.....

..... [2]

(b) Programming languages support different data types. These usually include `STRING` and `REAL`.

Complete the table by giving **four other** data types **and** an example data value for each.

Data type	Example data value

[4]

(c) An experienced programmer is working on a program that is written in a language she is not familiar with.

(i) State **one** feature of the program that she should be able to recognise.

.....

..... [1]

(ii) State the type of skill that would allow her to recognise this feature.

.....

..... [1]

(d) Give **three** methods that may be used to identify and locate errors in a program **after it has been written**.

You may include **one** feature found in a typical Integrated Development Environment (IDE).

1 .....

2 .....

3 .....

[3]

2 **(a)** An algorithm is needed to input a list of numbers representing test marks for a class of 30 students.
The algorithm will output the number of students who have a mark greater than 75. It will also output the average mark for the class.

Document the algorithm using **structured English**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(b) Each pseudocode statement in the following table contains an error.

State the error in each case.

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

Statement	Error
`Code ← LEFT(3, "Europe")`	
`Hour ← MID("ALARM:12:02", 7, 6)`	
`Size ← LENGTH(27.5)`	
`Num ← INT(27 / (Count + 3)`	
`Result ← "Conditional" AND "Loop"`	

[5]

(c) Part of a program flowchart is shown.

Write **program code** to implement the flowchart shown. Variable declarations are not required.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

3 A global 1D array, `ProdNum`, of type `INTEGER` contains 5 000 elements and is used to store product numbers.

A procedure is needed to sort `ProdNum` into ascending order using a bubble sort algorithm.

Write **program code** for the procedure `BubbleSort()`.

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]

4 **(a)** The following pseudocode includes a procedure that searches for a value in a 1D array and outputs each position in the array where the value is found.

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

```
DECLARE NameList : ARRAY [1:100] OF STRING
DECLARE SearchString : STRING

PROCEDURE Search()
DECLARE Index : INTEGER

FOR Index ← 1 TO 100
   IF NameList[Index] = SearchString
      THEN
         OUTPUT "Found at " & NUM_TO_STRING(Index)
   ENDIF
ENDFOR
ENDPROCEDURE
```

The specification of module `Search()` changes. The pseudocode needs to be amended to meet a new requirement.

The procedure needs to be implemented as a function, `Search()`, which will:

- take the search value as a parameter
- return an integer which is:
  - either the index value where the search value is **first** found
  - or `-1` if the search value is **not** found.

Write the **pseudocode** for the function `Search()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) A change to the specification in **part (a)** required a modification of the algorithm.

Give the term used for this type of modification.

..... [1]

(c) A change to the specification is only one reason to modify an algorithm.

Give **another** reason for the modification of an algorithm.

..... [1]

(d) Consider the following pseudocode:

```
10 DECLARE VarA : INTEGER
11 VarA ← 20
12
13 CALL ProcA(VarA)
14 OUTPUT VarA       // first value output
15
16 CALL ProcB(VarA)
17 OUTPUT VarA       // second value output
18
19
20 PROCEDURE ProcA(BYVALUE ThisValue : INTEGER)
21    ThisValue ← ThisValue + 5
22 ENDPROCEDURE
23
24 PROCEDURE ProcB(BYREF ThisValue : INTEGER)
25    ThisValue ← ThisValue + 5
26 ENDPROCEDURE
```

Procedures `ProcA()` and `ProcB()` use two methods of passing parameters.

Complete the following table.

	Output	**Explanation**
First value (line 14)	.....	....
Second value (line 17)	.....	....

[4]

(e) The procedures `ProcA` and `ProcB` in **part (d)** are examples of program modules.

Give **two** advantages of using program modules in program design.

1 .....

.....

2 .....

.....

[2]

5 A hashtag is used on a social media network to make it easier to find messages with a specific theme or content. A hashtag is a string consisting of a hash character '#' followed by a number of alphanumeric characters.

A message may contain several hashtag strings. A hashtag may be terminated by a space character, the start of the next hashtag, or by the end of the message.

For example, the following message contains three hashtags:

```
"#Alarm34 is the result of #BatteryFailure in the #PowerModule"
```

The hashtags in this message are `"#Alarm34"`, `"#BatteryFailure"` and `"#PowerModule"`.

A program is being developed to monitor their use.

The program will include two global arrays each containing 10 000 elements:

- A 1D array, `TagString`, of type `STRING` storing each hashtag in a single element of the array. All unused array elements contain an empty string (`""`).
- A 1D array, `TagCount`, of type `INTEGER` storing a count of the number of times each hashtag is used. The count value in a given element relates to the hashtag value stored in the element in the `TagString` array with the corresponding index value.

A developer has started to define the modules. Module `GetStart()` has already been written.

Module	Description

`GetStart()`	• Called with two parameters:
	  ◦ a message of type `STRING`
	  ◦ an integer giving the number of the required hashtag; for example, `GetStart(Message, 3)` would search for the third hashtag in the string `Message`
	• Returns an integer value representing the start position of the hashtag in the message, or value −1 if that hashtag does not exist

`AddHashtag()`	• Called with a hashtag of type `STRING` • Copies the hashtag to the next free element of the `TagString` array, and sets the corresponding element of the `TagCount` array to 1 • Returns `FALSE` if there are no unused elements in the `TagString` array, otherwise returns `TRUE`
`CountHashtag()`	• Called with a message of type `STRING` • Searches the message for hashtags using `GetStart()` • Returns a value representing the number of hashtags in the message
`IncrementHashtag()`	• Called with a hashtag of type `STRING` • Increments the value of the appropriate element in the `TagCount` array if the hashtag is found • Returns `TRUE` if the hashtag is found, or `FALSE` if the hashtag is not found

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

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

The module description is repeated here for reference.

Module	Description
`CountHashtag()`	• Called with a message of type `STRING` • Searches the message for hashtags using `GetStart()` • Returns a value representing the number of hashtags in the message

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

The module description is repeated here for reference.

Module	Description
`IncrementHashtag()`	• Called with a hashtag of type `STRING` • Increments the value of the appropriate element in the `TagCount` array if the hashtag is found • Returns `TRUE` if the hashtag is found, or `FALSE` if the hashtag is not found

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(d) A procedure, `OutputMostPop()`, is needed to output the most popular hashtag. The most popular hashtag is the one with the highest count value stored in the `TagCount` array.

As a reminder, the program includes two global arrays each containing 10 000 elements:

- A 1D array, `TagString`, of type `STRING` storing each hashtag in a single element of the array. All unused array elements contain an empty string (`""`).
- A 1D array, `TagCount`, of type `INTEGER` storing a count of the number of times each hashtag is used. The count value in a given element relates to the hashtag value stored in the element in the `TagString` array with the corresponding index value.

If the maximum count value occurs once, the procedure will output the corresponding hashtag and the count value.

It is possible for more than one hashtag to have the same highest count value. In this case, the procedure will output the maximum count value together with the number of hashtags with this maximum count value.

In both cases, the procedure must also output a suitable message.

You can assume that the arrays contain data for at least one hashtag.

Write **pseudocode** for the `OutputMostPop()` procedure.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

Appendix

Built-in functions (pseudocode)

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

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

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

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

`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 string `"BCD"`

`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"`

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`

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.

© UCLES 2020

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

3141164855

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2023**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (PQ/CB) 312089/3
© UCLES 2023

Refer to the **insert** for the list of pseudocode functions and operators.

1 The following pseudocode represents part of the algorithm for a program.

Line numbers are for reference only.

```
10   DECLARE Sheet4 : ARRAY[1:2, 1:50] OF INTEGER
…
100  FOR PCount ← 0 TO 49
101     Sheet4[1, PCount] ← 0
102     Sheet4[2, PCount] ← 47
103  NEXT PCount
```

(a) The pseudocode contains references to an array.

Complete the table by writing the answer for each row.

	Answer
The dimension of the array	
The name of the variable used as an array index	
The number of elements in the array	

[3]

(b) The pseudocode contains two errors. One error is that variable `PCount` has not been declared.

Identify the **other** error **and** state the line number where it occurs.

Error .....

.....

.....

Line number .....

[2]

(c) The pseudocode does not include a declaration for `PCount`.

State the data type that should be used in the declaration.

..... [1]

(d) The pseudocode statements given in the following table are used in other parts of the algorithm.

Complete the table by placing **one or more** ticks (✓) in each row.

The first row has already been completed.

Pseudocode statement	Input	Process	Output
`INPUT MyChoice`	✓		
`OUTPUT FirstName & LastName`			
`WRITEFILE YourFile, TextLine`			
`READFILE MyFile, TextLine`			
`Result ← SQRT(NextNum)`			

[4]

2 A program stores a date of birth for a student using a variable, `MyDOB`, of type `DATE`.

(a) `MyDOB` has been assigned a valid value corresponding to Kevin's date of birth.

Complete the pseudocode statement to test whether Kevin was born on a Thursday.

`IF` ..... `THEN` [2]

(b) A function `CheckDate()` will take three integer parameters representing a day, month and year of a given date.

The function will validate the date of birth for a student that the parameters passed to it represent.
For a date to be valid, a student must be at least 18 in year 2020.

(i) Two of the parameter values can be checked without reference to the third parameter.

Describe these **two** checks.

Check 1 .....

.....

.....

Check 2 .....

.....

.....

.....
[2]

(ii) Several values of the parameter representing the day can only be checked completely by referring to the value of **one other** parameter.

Describe this check.

.....

.....

.....

..... [2]

BLANK PAGE

3 A program processes data using a stack. The data is copied to a text file before the program ends.

(a) The following diagram shows the current state of the stack.

The operation of this stack may be summarised as follows:

- The `TopOfStack` pointer points to the last item added to the stack.
- The `BottomOfStack` pointer points to the first item on the stack.
- The stack grows upwards when items are added.

Stack		Pointer
Memory location	**Value**	
506		
505	WWW	← `TopOfStack`
504	YYY	
503	XXX	
502	ZZZ	
501	NNN	
500	PPP	← `BottomOfStack`

(i) An error will be generated if an attempt is made to POP a value when the stack is empty.

State the maximum number of consecutive POP operations that could be performed on the stack shown above **before** an error is generated.

..... [1]

(ii) The following operations are performed:

1. POP and store value in variable `Data1`
2. POP and store value in variable `Data2`
3. PUSH value AAA
4. PUSH value BBB
5. POP and discard value
6. POP and store value in variable `Data2`

Complete the diagram to show the state of the stack and the variables **after** the given operations have been performed.

Stack

Memory location	Value
506	
505	
504	
503	
502	
501	
500	

Pointer

Variable	Value
`Data1`	
`Data2`	

[4]

(b) The data is copied to a text file before the program ends.

(i) State an advantage of writing the data from the stack to a text file before the program ends.

.....

..... [1]

(ii) A module `SaveStack()` will write the data from the stack to a text file.

Express an algorithm for `SaveStack()` as five steps that could be used to produce pseudocode.

Write the **five** steps.

Step 1 .....

.....

Step 2 .....

.....

Step 3 .....

.....

Step 4 .....

.....

Step 5 .....

.....

[5]

4 A function `MakeString()` will:

1. take two parameters:
   - a count as an integer
   - a character
2. generate a string of length equal to the count, made up of the character
3. return the string generated, or return `"ERROR"` if the count is less than 1.

For example, the function call:

`MakeString(3, 'Z')` will return the string `"ZZZ"`

Write pseudocode for function `MakeString()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

5 A program is designed, coded and compiled without errors. The compiled code is sent for testing.

(a) The program will be tested using the walkthrough method.

Additional information will be needed before this method can be used.

Identify this additional information **and** explain why it is needed.

Additional information .....

.....

Explanation .....

.....

.....

.....
[3]

(b) Testing is completed and the program is made available to users.

Some time later, changes are made to the program to improve the speed of response.

State the type of maintenance that has been applied to the program.

..... [1]

6 A procedure `Select()` will:

1. take two integer values as parameters representing start and end values where both values are greater than 9 and the end value is greater than the start value
2. output **each** integer value between the start and the end value (**not** including the start and end values), where the sum of the last two digits is 6, for example, 142.

(a) Write pseudocode for procedure `Select()`.

Parameter validation is **not** required.

..... [7]

(b) The check performed by procedure `Select()` on the last two digits is needed at several places in the program and will be implemented using a new function.

The new function `CheckNum()` will:

- allow the required sum to be specified (not just 6)
- check one number
- return an appropriate value.

Describe the function interface **and two** advantages of this modular approach.

Interface .....

.....

.....

.....

Advantage 1 .....

.....

Advantage 2 .....

.....

[4]

7 A school has a library system which allows students to borrow books for a length of time. Information relating to students and books is stored in text files. Student information includes name, home address, email address, date of birth, tutor and subject choices. Book information includes author, title, subject category, library location and the date that the book was borrowed.

A program helps the staff to manage the borrowing of books.

(a) A new module needs to be written to generate emails to send to students who have an overdue book. Students who are sent an email are prevented from borrowing any more books until the overdue book is returned.

The process of abstraction has been used when designing the new module.

(i) State the purpose of applying abstraction to this problem.

.....

..... [1]

(ii) Identify **one** item of information that is required and **one** item that is **not** required in the new module. Justify your choices.

Item required .....

Justification .....

.....

.....

Item not required .....

Justification .....

.....

.....
[2]

(iii) Identify **two** operations that would be required to process data when an overdue book is returned.

Operation 1 .....

.....

Operation 2 .....

.....
[2]

(b) Part of the library program contains program modules with headers as follows:

Pseudocode module header
`PROCEDURE Module-X()`
`PROCEDURE Module-Y(BYREF RA : INTEGER, SA : REAL)`
`PROCEDURE Overlay()`
`FUNCTION Replace(RA : INTEGER, RB : BOOLEAN) RETURNS BOOLEAN`
`FUNCTION Reset(TA : STRING) RETURNS INTEGER`

`Module-X()` and `Module-Y()` are both called from module `Overlay()`.

Complete the structure chart.

[3]

8 A computer shop assembles desktop computers, using items bought from several suppliers. A text file `Stock.txt` contains information about each item.

Information for each item is stored as a single line in the `Stock.txt` file in the format:

```
<ItemNum><SupplierCode><Description>
```

Item information is as follows:

	Format	Comment
`ItemNum`	4 numeric characters	unique number for each item in the range “0001” to “5999” inclusive
`SupplierCode`	3 alphabetic characters	code to identify the supplier of the item
`Description`	a string	a minimum of 12 characters

The file is organised in ascending order of `ItemNum` and does not contain all possible values in the range.

The programmer has defined the first program module as follows:

Module	Description
`ChangeSupp()`	• called with two parameters `Code1` and `Code2` of type string that represent valid supplier codes • creates a new file `NewStock.txt` from the contents of the file `Stock.txt` where any reference to `Code1` is replaced by `Code2` • returns a count of the number of items that have had their supplier code changed

(a) Write pseudocode for module `ChangeSupp()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(b) A new module is required:

Module	Description

`Report_1()`	• takes a parameter of type string that represents a `SupplierCode`
	• searches the `Stock.txt` file for each line of item information that contains the given `SupplierCode`
	• produces a formatted report of items for the given `SupplierCode`, for example, for supplier DRG, the output could be:
	
	`Report for Supplier: DRG`
	
	`Item  Description`
	
	`1234  USB Printer Cable 3m`
	`1273  32GB USB Flash Drive`
	`1350  Mouse Mat 320 x 240mm`
	
	`Number of items listed: 3`

Write pseudocode for module `Report_1()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(c) The format of the output from module `Report_1()` from **part (b)** is changed. The number of items listed is moved to the top of the report as shown in the example:

```
Report for Supplier: DRG
Number of items listed: 3

Item        Description

1234        USB Printer Cable 3m
1273        32GB USB Flash Drive
1350        Mouse Mat 320 x 240mm
```

(i) Explain why this new layout would increase the complexity of the algorithm.

.....

.....

.....

..... [2]

(ii) The algorithm will be modified to produce the report in the new format. The modified algorithm will be implemented so that the file `Stock.txt` is only read once.

Describe the modified algorithm.

.....

.....

.....

.....

.....

.....

..... [3]

BLANK PAGE

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

8177307785

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2017**

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 **14** printed pages and **2** blank pages.

DC (JP) 154213
© UCLES 2017

CAMBRIDGE International Examinations

1 (a) (i) Procedural high-level languages usually support different data types.

Give an appropriate data type for each data value in the following table.

Data value	Data type
27	
"27"	
"27.3"	
TRUE	
27/3/2015	
27.3	

[6]

(ii) State an appropriate data structure to store the individual test scores for a class of students.

.....[1]

(iii) Describe how characters are represented using the ASCII character set.

.....

.....

.....

.....[2]

(b) Functions and procedures are subroutines.

Explain why you should use subroutines when designing a program solution.

.....

.....

.....

.....[2]

(c) The following pseudocode is an example of nested `IF` statements.

```
IF MyVar = 1
   THEN
      CALL Proc1()
   ELSE
      IF MyVar = 2
         THEN
            CALL Proc2()
         ELSE
            IF MyVar = 3
               THEN
                  CALL Proc3()
               ELSE
                  OUTPUT "Error"
            ENDIF
      ENDIF
ENDIF
```

Use **pseudocode** to write a `CASE` statement with the same functionality.

.....

.....

.....

.....

.....

.....

.....

.....[4]

(d) Program coding is a transferable skill.

You are given program code written in a high-level language that you have not studied.

State **two** different features of the code that you should be able to recognise.

1 .....

.....

2 .....

.....
[2]

2 The following pseudocode represents a simple algorithm.

```
DECLARE NumberFound, Remainder, Number : INTEGER
DECLARE StartNumber, EndNumber, Divisor : INTEGER

INPUT StartNumber
INPUT EndNumber
INPUT Divisor
NumberFound ← 0

FOR Number ← StartNumber TO EndNumber
   Remainder ← MODULUS(Number, Divisor)
   IF Remainder = 0
      THEN
         OUTPUT Number
         NumberFound ← NumberFound + 1
   ENDIF
ENDFOR
OUTPUT "Count: " & NumberFound
```

For the built-in functions list, refer to the **Appendix** on page 14.

(a) Complete the following trace table.

StartNumber	EndNumber	Divisor	NumberFound	Number	Remainder	Output
11	13	2	0			

[3]

(b) Describe the purpose of this algorithm.

.....

.....

.....

.....

.....

.....[3]

(c) Draw a program flowchart to represent this algorithm.

Variable declarations are **not** required in program flowcharts.

[10]

3 **(a)** A multi-user computer system stores information about users. It uses a 1D array, `UserNameArray`, of type `STRING`. There are 100 elements in the array.

The format of the string in each element of the array is as follows:

```
<UserID><UserName>
```

- `UserID` is a six-character string of numerals.
- `UserName` is a variable-length string.

Write **pseudocode** for a procedure, `BubbleSort`, to perform an efficient bubble sort on `UserNameArray`. The array is to be sorted in ascending order of `UserID`.

You should assume that `UserNameArray` has been declared as a global variable.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

(b) The value of `UserID` should be unique for each user but a problem has occurred and repeated `UserID` values may have been issued.

The array is sorted by `UserID`, so any repeated `UserID` values will appear in consecutive array elements.

A procedure, `FindRepeats` is required.

This will:

- compare each element with the previous element and output the `UserID` and `UserName` if the `UserID` is repeated
- output the total number of `UserID`s that are repeated.

For example, the `UserNameArray` contains the following entries.

Array element	Comment
⋮	
`122222Jim Moriarty`	
⋮	
`123456Fred Smith`	
`123456Eric Sykes`	Repeated User ID
`123456Kevin Turvey`	Repeated User ID
⋮	
`222244Alice Chan`	
`222244Myra Singh`	Repeated User ID
⋮	
`333333Yasmin Halim`	
⋮	

For this example, the output is:

```
123456Eric Sykes
123456Kevin Turvey
222244Myra Singh
There are 3 repeated UserIDs
```

If no repeated `UserID`s are found, the output is:

```
The array contains no repeated UserIDs
```

Write **program code** for the procedure, `FindRepeats`.
You should assume that `UserNameArray` has been declared as a global variable.

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]

(c) **(i)** The `FindRepeats` procedure forms part of a program.

Name **three** stages in a program development cycle.

1 .....

2 .....

3 .....
[3]

(ii) The program containing `FindRepeats` will be created using an IDE.

State what is meant by IDE.

.....

.....[1]

(iii) Name **two** features provided by an IDE that assist in the program development cycle.

1 .....

.....

2 .....

.....
[2]

(iv) The procedure, `FindRepeats`, is written assuming there are 100 elements in `UserNameArray`.

In the main program, the global array, `UserNameArray`, has been declared with only 50 elements.

State the type of error this will cause.

.....[1]

4 Numeric formatting converts a numeric value to a string in order to present it in a specific way.

In a generic high-level language, formatting is implemented using a mask system. In this system, each character of the mask corresponds to one character of the formatted string.

Mask characters have the following meaning:

Mask character	Meaning
#	Character must be a digit or a space
0	Character must be a digit

Any other mask characters are taken as literal values and are included in the formatted string.

(a) Using the mask `"###00.00"`, complete the following table. Use □ to represent a space. The first value has been done for you.

Value	Formatted string
1327.5	"□1327.50"
1234	
7.456	

[2]

(b) For each row in the following table, define the mask required to produce the formatted output from the given value. □ represents a space.

Value	Required output	Mask
1234.00	"1,234.00"	
3445.66	"£3,445.66"	
10345.56	"$□□10,345"	

[3]

5 A sports club maintains a record of the email address of each of its members. The details are stored in a text file, `EmailDetails.txt`. The format of each line of the text file is as follows:

```
<MembershipNumber><EmailAddress>
```

- `MembershipNumber` is a four-character string of numerals.
- `EmailAddress` is a variable-length string.

Membership of the club has increased and a four-character membership number is no longer adequate.

A procedure, `MakeNewFile`, is required to perform the following actions:

1. Create a new file, `NewEmailDetails.txt`
2. Read a line from file `EmailDetails.txt`
3. Extend `MembershipNumber` by adding two leading zero digits (for example, `"1234"` becomes `"001234"`)
4. Write the new line to file `NewEmailDetails.txt`
5. Repeat steps 2 to 4 for all lines in the original file.

(a) Write **pseudocode** for the procedure `MakeNewFile`.

For the built-in functions list, refer to the **Appendix** on page 14.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

(b) A function, `IsEmailValid`, is to be written to test for a valid email address format.

An email address has a valid format if it obeys the following three rules:

1. It contains a single '@' symbol.
2. The '@' symbol must be preceded by at least one character.
3. The '@' symbol must be followed by at least three characters.

Choose **three** different invalid strings to test distinct aspects of the rules.

Explain your choice in each case.

1 .....

Explanation .....

.....

.....

2 .....

Explanation .....

.....

.....

3 .....

Explanation .....

.....

.....

[6]

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

```
MODULUS(x : INTEGER, y : INTEGER) RETURNS INTEGER
```

returns the remainder when `x` is divided by `y` using integer arithmetic.
Example: `MODULUS(5, 2)` returns `1`

```
INT(x : REAL) RETURNS INTEGER
```

returns the integer part of `x`.
Example: `INT(27.5415)` returns `27`

```
LENGTH(ThisString : STRING) RETURNS INTEGER
```

returns the integer value representing the length of string `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 string `"ABC"`

```
RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING
```

returns rightmost `x` characters from `ThisString`.
Example: `RIGHT("ABCDEFGH", 3)` returns string `"FGH"`

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`

© UCLES 2017

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2017

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

0638561535

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2020**

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. Blank pages are indicated.

DC (CE/CB) 180782/4
© UCLES 2020

1 **(a)** Complete this definition of the term **algorithm**.

An algorithm is a solution to a problem expressed as .....

.....

..... [2]

(b) A program design includes the use of subroutines (functions and procedures).

Give **three** advantages of using subroutines in a program.

1 .....

.....

2 .....

.....

3 .....

.....
[3]

(c) Draw lines on the following diagram to connect each computing term with the appropriate description.

Term	Description
Selection	Checking that a program performs as expected
Black-box testing	A method for increasing the level of detail of an algorithm
Stepwise refinement	To test a condition to determine the path of program execution
Iteration	A method of executing certain lines of code more than once

[3]

2 **(a)** Three modules form part of a program for a car rental company. A description of the relationship between the modules is summarised as follows:

Module name	Description
`RentCar()`	A customer will pay for each car rental either by bank card or by using their account with the rental company.
`PayByCard()`	Called with parameter `HireCost`, representing the cost of the rental. Returns a `BOOLEAN` value to indicate whether or not the card payment was successful.
`PayByAccount()`	Called with parameters `HireCost`, `AccountNumber`, `CurrentBalance` and `AccountLimit`. • Checks whether `HireCost` plus the `CurrentBalance` would exceed the `AccountLimit`. If so, then the rental is not authorised. • If the rental is authorised, then the `CurrentBalance` is updated. • Returns a `BOOLEAN` value to indicate whether or not the rental was authorised.

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

[5]

(b) The following pseudocode algorithm has been developed to check whether a string contains a valid password.

To be a valid password, a string must:

- be longer than 6 characters
- contain at least one lower case letter
- contain at least one upper case letter
- contain at least one non-alphabetic character.

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

  DECLARE Index : INTEGER
  DECLARE StrLen : INTEGER
  DECLARE NumUpper, NumLower : INTEGER
  DECLARE NumNonAlpha : INTEGER
  DECLARE NextChar : CHAR

  NumUpper ← 0
  NumLower ← 0
  NumNonAlpha ← 0

  StrLen ← LENGTH(InString)
  IF StrLen < 7
     THEN
        RETURN FALSE
     ELSE
        FOR Index ← 1 TO StrLen
           NextChar ← MID(InString, Index, 1)
           IF NextChar >= 'a' AND NextChar <= 'z'
              THEN
                 NumLower ← NumLower + 1
              ELSE
                 IF NextChar > 'A' AND NextChar <= 'Z'
                    THEN
                       NumUpper ← NumUpper + 1
                    ELSE
                       NumNonAlpha ← NumNonAlpha + 1
                 ENDIF
           ENDIF
        ENDFOR
  ENDIF

  IF (NumUpper >= 1) AND (NumLower >= 1) AND (NumNonAlpha >= 1)
     THEN
        RETURN TRUE
     ELSE
        RETURN FALSE
  ENDIF

ENDFUNCTION
```

Refer to the **Appendix** on page 21 for a list of built-in pseudocode functions and operators.

The pseudocode does not work under all circumstances.

A dry run performed on the function `Check()`, with the string `"crAsh99"`, produced the following trace table.

The string is a valid password, but the pseudocode would return the value `FALSE`.

Trace table row	StrLen	Index	NextChar	NumUpper	NumLower	NumNonAlpha
1	7			0	0	0
2		1	'c'			
3					1	
4		2	'r'			
5					2	
6		3	'A'			
7						1
8		4	's'			
9					3	
10		5	'h'			
11					4	
12		6	'9'			
13						2
14		7	'9'			
15						3

(i) Describe how the completed trace table may be used to identify the error in the pseudocode. In your answer, refer to the trace table row number(s).

.....

.....

.....

.....

..... [2]

(ii) State the **pseudocode** line number that has to be changed to correct the error **and** write the correct pseudocode for the complete line.

Line number .....

Correct pseudocode .....

.....
[2]

(iii) Rewrite lines 29 to 39 of the original pseudocode using a `CASE` structure.

.....

.....

.....

.....

.....

.....

..... [4]

Question 3 begins on the next page.

3 **(a)** A mobile phone provider has developed an account management program. The program includes a procedure, `AddCredit()`. The procedure is called with two parameters, `TopUp` and `PhoneNum`.

The relevant part of the identifier table and the program flowchart for the procedure are as shown:

Identifier	Type	Description
`TopUp`	`REAL`	The amount of credit to be added
`PhoneNum`	`STRING`	The unique customer phone number
`Balance`	`REAL`	The current amount of credit
`Multiple`	`REAL`	The amount of credit bonus
`GetBalance()`	`FUNCTION`	Takes the phone number as a parameter and returns the current balance
`SetBalance()`	`PROCEDURE`	Takes the phone number and the new balance as parameters and updates the account with the new balance

Write **pseudocode** to implement the procedure `AddCredit()`. The pseudocode must follow the algorithm represented by the flowchart. Declare any local variables used.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) The following pseudocode searches for a string `"Chris"` in a 1D array and outputs the index positions where the string is found.

```
DECLARE NameList : ARRAY [1:100] OF STRING
DECLARE n : INTEGER

FOR n ← 1 TO 100
   IF NameList[n] = "Chris"
      THEN
         OUTPUT "Found at: " & NUM_TO_STRING(n)
   ENDIF
ENDFOR
```

The pseudocode needs to be modified as follows:

- Write the search as a procedure, `Search()`, that takes the search string as a parameter.
- Change the array to a 2D array. The first dimension contains names and the second dimension contains the corresponding status.

  For example:

  ```
  NameList[23, 1] ← "Chris"        // name
  NameList[23, 2] ← "On Holiday"   // status
  ```

- Detect a match only when the name contains the search string **and** the status contains `"Active"`.
- If a match has been detected, the procedure will output a **single** message giving all of the index positions where a match occurred. For example, `"Found at: 3 6 22"`.
- If no match has been detected, the procedure will output a suitable message.

Refer to the **Appendix** on page 21 for a list of built-in pseudocode functions and operators.

Write the **pseudocode** for the procedure `Search()`. Assume the array has been declared globally.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

4 **(a)** An inexperienced user buys a games program. A program fault occurs while the user is playing the game.

Explain what is meant by a **program fault**.

.....

.....

.....

.....

.....

.....

..... [2]

(b) Give **three** ways to minimise the risk of faults when writing programs.

1 .....

.....

2 .....

.....

3 .....

..... [3]

(c) Three types of program error are syntax, logic and run-time.

Define these **three** types.

Syntax error .....

.....

.....

Logic error .....

.....

.....

Run-time error .....

.....

.....
[3]

Question 5 begins on the next page.

5 **(a)** A 1D array, `Directory`, of type `STRING` is used to store a list of school internal telephone numbers. There are 1000 elements in the array. Each element stores a single data item. The format of each data item is as follows:

```
<Number><Name>
```

`Number` is a four-digit numeric string.
`Name` is a variable-length string.

For example:

```
"1024Collins Robbie"
```

The following pseudocode is an initial attempt at defining a procedure `SortContacts()` that will perform a bubble sort on `Directory`. The array is to be sorted in ascending order of `Name`.

Fill in the gaps to complete the pseudocode.

Refer to the **Appendix** on page 21 for a list of built-in pseudocode functions and operators.

```
PROCEDURE SortContacts ()
DECLARE Temp : STRING
DECLARE FirstName, SecondName : STRING
DECLARE NoSwaps : ........................................
DECLARE Boundary, J : INTEGER
Boundary ← ........................
REPEAT
   NoSwaps ← TRUE
   FOR J ← 1 TO Boundary
      FirstName ← ..............................(Directory[J], LENGTH(Directory[J]) - ............... )
      SecondName ← RIGHT(Directory[J + 1], LENGTH(Directory[J + 1]) - 4)
      IF FirstName ....................................
        THEN
          Temp ← Directory[J]
          Directory[J] ← Directory ...............................
          Directory[J + 1] ← Temp
          NoSwaps ← ...............................
      ENDIF
   ENDFOR
   Boundary ← ...............................
UNTIL NoSwaps = TRUE
ENDPROCEDURE
```

[8]

(b) The pseudocode contains a mechanism designed to make this an efficient bubble sort.

Describe the mechanism **and** explain why it may be considered to be efficient.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

6 A company hires out rowing boats on a lake. The company has ten boats, numbered from 1 to 10.

The company is developing a program to help manage and record the hiring out process.

Hire information is stored in three global 1D arrays when a boat is hired out. Each array contains 10 elements representing each of the ten boats.

The three 1D arrays are summarised as follows:

Array	Data type	Description	Example data value
`HireTime`	`STRING`	The time the boat was hired out	`"10:15"`
`Duration`	`INTEGER`	The number of minutes of the hire	`30`
`Cost`	`REAL`	The cost of the hire	`5.75`

If an individual boat is not currently on hire, the corresponding element of the `HireTime` array will be set to `"Available"`.

The programmer has started to define program modules as follows:

Module	Description

`AddTime()`	• Called with two parameters:
	  ○ a `STRING` value representing a time
	  ○ an `INTEGER` value representing a duration in minutes
	• Adds the duration to the time to give a new time
	• Returns the new time as a `STRING`

`ListAvailable()`	• Called with a `STRING` value representing the time the hire will start
	• Outputs the boat numbers that will be available for hire at the given start time. A boat will be available for hire if it is either:
	  ○ currently not on hire, or
	  ○ due back before the given hire start time
	• Outputs the number of each boat available
	• Outputs the total number of boats available or a suitable message if there are none

`RecordHire()`	• Called with four parameters:
	  ○ an `INTEGER` value representing the boat number
	  ○ a `STRING` value representing the hire start time
	  ○ an `INTEGER` value representing the hire duration in minutes
	  ○ a `REAL` value representing the cost of hire
	• Updates the appropriate element in each array
	• Adds the cost of hire to the global variable `DailyTakings`
	• Converts the four input parameters to strings, concatenated using commas as separators, and writes the resulting string to the end of the existing text file `HireLog.txt`

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

.....

.....

.....

.....

..... [8]

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

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

Module	Description

`AddTime()`	• Called with two parameters:
	  ○ a `STRING` value representing a time
	  ○ an `INTEGER` value representing a duration in minutes
	• Adds the duration to the time to give a new time
	• Returns the new time as a `STRING`

(i) Write **program code** for a statement that uses the `AddTime()` function to add a duration of 60 minutes to a start time contained in variable `BeginTime` and to assign the new time to variable `EndTime`.

Programming language .....

Program code

.....

..... [2]

(ii) The function `AddTime()` is to be tested using black-box testing.

Complete the following **two** tests that can be performed to check the operation of the function. Note that test 1 and test 2 are different.

Test 1 – Boat is returned during the same hour as rental starts

Start time value ..... Duration value .....

Expected new time value .....

Test 2 – Boat is returned during the hour after the rental starts

Start time value ..... Duration value .....

Expected new time value .....

[2]

Appendix

Built-in functions (pseudocode)

```
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"`

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

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.

© UCLES 2020

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

0910143936

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2021**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

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

DC (PQ) 214717
© UCLES 2021

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** A program is being developed to help manage the membership of a football club.

Complete the following identifier table.

Example value	Explanation	Variable name	Data type
`"Wong"`	The preferred name of the member joining the football club		
`FALSE`	A value to indicate whether an existing member of the club lives at the same address		
`19/02/1983`	When the member joined the football club		
`1345`	The number of points a member has earned. Members of the club earn points for different activities.		

[4]

(b) 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.

You can assume that none of the variables referenced are of an incorrect type.

Statement	Error
`Result ← 2 & 4`	
`SubString ← MID("pseudocode", 4, 1)`	
`IF x = 3 OR 4 THEN`	
`Result ← Status AND INT(x/2)`	
`Message ← "Done" + LENGTH(MyString)`	

[5]

(c) The following data items need to be stored for each student in a group:

- student name (a string)
- test score (an integer).

State a suitable data structure and justify your answer.

Structure .....

Justification .....

.....

.....
[3]

2 **(a)** Four program modules form part of a program for a library.

A description of the relationship between the modules is summarised as follows:

Module name	Description
`UpdateLoan()`	• Calls either `LoanExtend()` or `LoanReturn()`
`LoanExtend()`	• Called with parameters `LoanID` and `BookID` • Calls `CheckReserve()` to see whether the book has been reserved for another library user • Returns `TRUE` if the loan has been extended, otherwise returns `FALSE`
`CheckReserve()`	• Called with `BookID` • Returns `TRUE` if the book has been reserved, otherwise returns `FALSE`
`LoanReturn()`	• Called with parameters `LoanID` and `BookID` • Returns a `REAL` (which is the value of the fine to be paid in the case of an overdue loan)

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

[5]

(b) The definition for module `LoanReturn()` is amended as follows:

Module name	Description
`LoanReturn()`	Called with parameters `LoanID`, `BookID` and `Fine` The module code checks whether the book has been returned on time and then assigns a new value to `Fine`

- `LoanID` and `BookID` are of type `STRING`
- `Fine` is of type `REAL`

Write the pseudocode header for the **amended** module `LoanReturn()`.

.....

..... [2]

(c) A program will:

- input 50 unique integer values
- output the largest value
- output the average of the values **excluding** the largest value.

Draw a program flowchart to represent the algorithm.

Variable declarations are **not** required.

It is not necessary to check that each input value is unique.

[6]

BLANK PAGE

3 **(a)** A concert venue uses a program to calculate admission prices and store information about ticket sales.

A number of arrays are used to store data. The computer is switched off overnight and data has to be input again at the start of each day before any tickets can be sold. This process is very time consuming.

(i) Explain how the program could use text files to speed up the process.

.....

.....

.....

.....

.....

..... [2]

(ii) State the characteristic of text files that allow them to be used as explained in **part (a)(i)**.

.....

..... [1]

(iii) Information about ticket sales will be stored as a booking. The booking requires the following data:

- name of person booking
- number of people in the group (for example a family ticket or a school party)
- event type.

Suggest how data relating to each booking may be stored in a text file.

.....

.....

.....

..... [2]

(b) A procedure `Preview()` will:

- take the name of a text file as a parameter
- output a warning message if the file is empty
- otherwise output the first five lines from the file (or as many lines as there are in the file if this number is less than five).

Write pseudocode for the procedure `Preview()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

4 Study the following pseudocode. Line numbers are for reference only.

```
10 FUNCTION Convert(Name : STRING) RETURNS STRING
11
12   DECLARE Flag: BOOLEAN
13   DECLARE Index : INTEGER
14   DECLARE ThisChar : CHAR
15   DECLARE NewName : STRING
16
17   CONSTANT SPACECHAR = ' '
18
19   Flag ← TRUE
20   Index ← 1
21   NewName ← ""      // formatted name string
22
23   WHILE Index <= LENGTH(Name)
24      ThisChar ← MID(Name, Index, 1)
25      IF Flag = TRUE THEN
26         NewName ← NewName & UCASE(ThisChar)
27         IF ThisChar <> SPACECHAR THEN
28            Flag ← FALSE
29         ENDIF
30      ELSE
31         NewName ← NewName & ThisChar
32      ENDIF
33      IF ThisChar = SPACECHAR THEN
34         Flag ← TRUE
35      ENDIF
36      Index ← Index + 1
37   ENDWHILE
38
39   RETURN NewName
40
41 ENDFUNCTION
```

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

```
Result ← Convert("∇in∇a∇∇Cup")
```

Note: The symbol `'∇'` has been used to represent a space character.
Use this symbol for any space characters in the trace table.

The first row has been completed for you.

Name	Flag	Index	NewName	ThisChar
"∇in∇a∇∇Cup"				

[5]

(b) The pseudocode for `Convert()` contains a conditional loop.

State a more appropriate loop structure.

Justify your answer.

Loop structure .....

.....

Justification .....

.....

.....
[2]

(c) Two changes need to be made to the algorithm.

Change 1: Convert to lower case any character that is not the first character after a space.
Change 2: Replace multiple spaces with a single space.

(i) Change 1 may be implemented by modifying one line of the pseudocode.

Write the modified line.

.....

.....[1]

(ii) Change 2 may be implemented by moving one line of the pseudocode.

Write the number of the line to be moved and state its new position.

Line number .....

New position .....

.....
[2]

BLANK PAGE

5 A global 2D array `Result` of type `INTEGER` is used to store a list of exam candidate numbers together with their marks. The array contains 2000 elements, organised as 1000 rows and 2 columns.

Column 1 contains the candidate number and column 2 contains the mark for the corresponding candidate. All elements contain valid exam result data.

A procedure `Sort()` is needed to sort `Result` into ascending order of mark using an efficient bubble sort algorithm.

Write pseudocode for the procedure `Sort()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

6 The following diagram represents an Abstract Data Type (ADT) for a linked list.

The free list is as follows:

(a) Explain how a node containing data value B is added to the list in alphabetic sequence.

.....

.....

.....

.....

.....

.....

..... [4]

(b) Describe how the linked list in **part (a)** may be implemented using variables and arrays.

.....

.....

.....

..... [2]

BLANK PAGE

7 A program is needed to take a string containing a full name and produce a new string of initials.

Some words in the full name will be ignored. For example, "the", "and", "of", "for" and "to" may all be ignored.

Each letter of the abbreviated string must be upper case.

For example:

Full name	Initials
Integrated Development Environment	IDE
The American Standard Code for Information Interchange	ASCII

The programmer has decided to use a global variable `FNString` of type `STRING` to store the full name.

It is assumed that:

- words in the full name string are separated by a single space character
- space characters will not occur at the beginning or the end of the full name string
- the full name string contains at least one word.

The programmer has started to define program modules as follows:

Module	Description
`GetStart()`	• Called with an `INTEGER` as a parameter, representing the number of a word in `FNString`. • Returns the character start position of that word in `FNString` or returns `-1` if that word does not exist • For example: if `FNString` contains the string `"hot and cold"`, `GetStart(3)` returns `9`
`GetWord()`	• Called with a parameter representing the position of the first character of a word in `FNString` • Returns the word from `FNString` • For example: if `FNString` contains the string `"hot and cold"`, `GetWord(9)` returns `"cold"`

(a) Write pseudocode for the module `GetStart()`.

.....

.....

.....

.....

.....

.....

..... [7]

(b) The programmer has decided to use a global ten-element 1D array `IgnoreList` of type `STRING` to store the ignored words. Unused elements contain the empty string (`""`) and may occur anywhere in the array.

A new module `AddWord()` is needed as follows:

Module	Description
`AddWord()`	• Called with a parameter representing a word • Stores the word in an unused element of the `IgnoreList` array and returns `TRUE` • Returns `FALSE` if the array was already full or if the word was already in the array

Write a detailed description of the algorithm for `AddWord()`. Do **not** include pseudocode statements in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) As a reminder, the module description of `GetWord()` is repeated:

Module	Description
`GetWord()`	• Called with a parameter representing the position of the first character of a word in `FNString` • Returns the word from `FNString` • For example: if `FNString` contains the string `"hot and cold"`, `GetWord(9)` returns `"cold"`

Write pseudocode for the module `GetWord()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

BLANK PAGE

BLANK PAGE

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2022**

INSERT **2 hours**

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (EF) 311804
© UCLES 2022

Functions

Note: An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

`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"`
`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 string `"BCD"`
`LENGTH(ThisString : STRING) RETURNS INTEGER` returns the integer value representing the length of string `ThisString` Example: `LENGTH("Happy Days")` returns `10`
`LCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the lower-case equivalent of `ThisChar` Alphabetic characters that are not upper case are returned unchanged. Example: `LCASE('W')` returns `'w'`
`UCASE(ThisChar : CHAR) RETURNS CHAR` returns the character representing the upper-case equivalent of `ThisChar` Alphabetic characters that are not lower case are returned unchanged. Example: `UCASE('a')` returns `'A'`
`TO_UPPER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to upper case. Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`
`TO_LOWER(ThisString : STRING) RETURNS STRING` returns a string formed by converting all characters of `ThisString` to lower case. Example: `TO_LOWER("JIM 803")` returns `"jim 803"`
`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>` returns a string representation of a numeric value. Note: `<datatype1>` may be `REAL` or `INTEGER`, <datatype2> may be `CHAR` or `STRING` Example: `NUM_TO_STR(87.5)` returns `"87.5"`
`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>` returns a numeric representation of a string. Note: `<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER` Example: `STR_TO_NUM("23.45")` returns `23.45`

String and Character Functions

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN` returns the value `TRUE` if `ThisString` represents a valid numeric value. Note: `<datatype>` may be `CHAR` or `STRING` Example: `IS_NUM("-12.36")` returns `TRUE`
`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`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`

Date Functions

Note: Date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2, etc. Example: `DAYINDEX(12/05/2020)` returns `3`
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a variable of type `DATE` with the value of `<Day>/<Month>/<Year>`
`TODAY() RETURNS DATE` returns a variable of type `DATE` with the value set to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: The function will generate an error if the file is not already open in `READ` mode.

Operators

Note: An error will be generated if an operator is used with a value or values of an incorrect type.

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: May also be used to concatenate a `CHAR` with a `STRING`
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	Used to compare two items of the same type. Returns `TRUE` if the condition is true, otherwise returns `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records.
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `count = 4` evaluates to `TRUE` when variable `count` contains the value `4`

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2022

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2020**

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 **20** pages. Blank pages are indicated.

DC (CJ) 188580/2
© UCLES 2020

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

One stage is INPUT.

Name the **other** stages.

1 .....

2 .....
[1]

(b) An algorithm may be documented using different methods. These include structured English, a program flowchart, and pseudocode.

State what a program designer represents using one or more of these methods.

.....

..... [2]

(c) Programming languages support different data types.

Complete the table by giving four **different** data types together with an example data value for each.

Data type	Example data value

[4]

(d) Draw lines to connect each of the following computing terms with the appropriate description.

Term	Description
Black-box testing	A structure for the temporary storage of data
File	A method used when the structure of the program is unknown
Assignment	A method of setting the value of a variable
Array	A structure for the permanent storage of data

[3]

(e) A pseudocode algorithm assigns values to three variables as follows:

```
FlagA ← TRUE
FlagB ← FALSE
FlagC ← TRUE
```

Evaluate the expressions given in the following table:

Expression	Evaluates to
`NOT FlagB AND FlagC`	
`NOT (FlagB OR FlagC)`	
`(FlagA AND FlagB) OR FlagC`	
`NOT (FlagA AND FlagB) OR NOT FlagC`	

[2]

2 **(a)** The following pseudocode is an attempt to define an algorithm that takes two numbers as input and outputs the larger of the two numbers.

```
DECLARE A, B : INTEGER
INPUT A
INPUT B
IF A > B
   THEN
      OUTPUT A
   ELSE
      OUTPUT B
ENDIF
```

The algorithm needs to be amended to include the following changes:

1. Input **three** values, ensuring that each value input is unique.
2. Output the average.
3. Output the largest value.

Write the **pseudocode** for the amended algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) Complete the pseudocode expressions in the following table.

Use **only** functions and operators described in the **Appendix** on pages 18–19.

Expression	Evaluates to
`"ALARM: " &` ..... `("Time: 1202" ,` .....`)`	`"ALARM: 1202"`
..... `("Stepwise." ,` ..... `,` .....`)`	`"wise"`
`1.5 *` ..... `("OnePointFive")`	`18`
..... `(27.5)`	`"27.5"`
..... `(9, 4)`	`2`

[5]

(c) A problem may be decomposed into sub-tasks when designing an algorithm.

Give **three** benefits of using sub-tasks.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

3 A car has the ability to detect a skid by monitoring the rate of rotation (the rotational speed) of each wheel. If the rate of rotation of any wheel is not within 10% of the average of all four wheels, the car skids.

A function, `CheckSkid()`, is being developed.

The function will:

- simulate real-time data acquisition, by prompting for the input of four integer values in the range 0 to 1000 inclusive, representing the rate of rotation of each wheel
- calculate the average value
- check whether any individual value is more than 10% greater than the average or more than 10% less than the average
- return `TRUE` if any individual value is more than 10% greater than the average or more than 10% less than the average and `FALSE` otherwise
- output a suitable warning message.

(a) Write **program code** for the function `CheckSkid()`.

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) Give **two** sets of test data that could be used to test the function.

Test 1 – No skid detected

Value1	Value2	Value3	Value4

Test 2 – Skid detected

Value1	Value2	Value3	Value4

[2]

4 **(a)** The following structured English describes an algorithm used to count the number of odd and even digits in an input sequence.

1. Initialise variables `OddCount` and `EvenCount` to zero.
2. Prompt and input an integer.
3. If the integer is not in the range 0 to 9 then go to step 7.
4. If the integer is an even number then add 1 to `EvenCount`.
5. Otherwise add 1 to `OddCount`.
6. Repeat from step 2.
7. Output `"Same"` if there are the same number of odd and even integers.
8. Output `"Odd"` if there are more odd than even integers.
9. Output `"Even"` if there are more even than odd integers.

Draw a flowchart on the following page to represent the algorithm.

[7]

(b) The following pseudocode is an attempt to check whether two equal-length strings consist of identical characters.

Refer to the **Appendix** on pages 18–19 for the list of built-in functions and operators.

```
FUNCTION Compare(String1, String2 : STRING) RETURNS BOOLEAN
   DECLARE x, y, Len1, Len2 : INTEGER
   DECLARE RetFlag : BOOLEAN
   DECLARE NextChar : CHAR
   DECLARE New : STRING

   Len1 ← LENGTH(String1)
   RetFlag ← TRUE

   FOR x ← 1 TO Len1                          // for each char in String1
      Len2 ← LENGTH(String2)
      NextChar ← MID(String1, x, 1)           // get NextChar from String1
      New ← ""
      FOR y ← 1 TO Len2                       // for each char in String2
         IF NextChar <> MID(String2, y, 1)     // no match
            THEN
               New ← New & MID(String2, y, 1) // save this char from String2
         ENDIF
      ENDFOR
      String2 ← New                           // replace String2 with New
   ENDFOR

   IF LENGTH(String2) <> 0                    // anything left in String2 ?
      THEN
         RetFlag ← FALSE
   ENDIF

   RETURN RetFlag

ENDFUNCTION
```

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

```
Result ← Compare("SUB", "BUS")
```

The first row has been completed for you.

String1	String2	Len1	RetFlag	x	Len2	NextChar	New	y
"SUB"	"BUS"	3	TRUE	1				

[5]

(ii) State the value returned.

..... [1]

(iii) There is an error in the algorithm, which means that under certain circumstances, the function will return an incorrect value.

Describe the problem. Give **two** test strings that would demonstrate it.

Problem .....

.....

.....

.....

.....

Test String1 .....

Test String2 .....
[2]

(iv) Describe the modification that needs to be made to the algorithm to correct the error.

Do **not** use pseudocode or program code in your answer.

.....

.....

.....

.....

..... [1]

(v) State the name given to the type of testing that makes use of a trace table.

..... [1]

(vi) State **two** features found in a typical Integrated Development Environment (IDE) that may be used for debugging a program.

1 .....

2 .....
[2]

Question 5 begins on the next page.

5 A hashtag is used on a social media network. A hashtag is a string consisting of a hash character '#' followed by one or more alphanumeric characters.

A program is being developed to monitor the use of hashtags.

The program will include two global arrays each containing 10 000 elements:

- A 1D array, `TagString`, of type `STRING` stores each hashtag in a single element. All unused array elements contain an empty string (`""`).
- A 1D array, `TagCount`, of type `INTEGER` stores a count of the number of times each hashtag is used. The count value at a given index relates to the element stored at the corresponding index in the `TagString` array.

The contents of the two arrays will be stored in a text file `Backup.txt`. The format of each line of the file is:

```
<Hashtag><','><Count>
```

For example:

```
"#ComputerScienceClass,978"
```

A developer has started to define the modules as follows:

Module	Description
`InitArrays()`	• Initialise the arrays
`SaveArrays()`	• The contents of the two arrays are stored in the text file `Backup.txt` Existing file contents will be overwritten • Each hashtag and count are stored in one line of the file, as in the example above • Unused `TagString` elements are not added to the file • Returns the total number of unused `TagString` elements
`LoadArrays()`	• Values from the text file `Backup.txt` are stored in the two arrays • The number of elements stored is returned

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(c) Write **program code** for the module `LoadArrays()`.

The module description is repeated here for reference.

Module	Description
`LoadArrays()`	• Values from the text file `Backup.txt` are stored in the two arrays • The number of elements stored is returned

You should assume:

- each line of the file contains a string of the correct format and no validation checks are required
- there are no more than 10 000 lines in the file.

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]

Appendix

Built-in functions (pseudocode)

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

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

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

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

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

`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 string `"BCD"`

`DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`

Example: `DIV(10,3)` returns `3`

`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`

© UCLES 2020

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`

© UCLES 2020

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.

© UCLES 2020

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 4 6 3 0 1 0 8 0 0 4 *

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2015**

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 **16** printed pages.

DC (NH/CGW) 95376/4
© UCLES 2015

CAMBRIDGE International Examinations

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

1 Computer programs have to evaluate expressions.

Study the sequence of pseudocode statements.

Give the value assigned to each variable.

The statement may generate an error. If so, write ERROR.

The & operator is used to concatenate strings.

```
DECLARE N1        : INTEGER
DECLARE N2        : INTEGER
DECLARE Answer    : REAL
DECLARE Found     : BOOLEAN
DECLARE IsValid   : BOOLEAN
N1 ← 3
N2 ← 9
Answer ← (N1 + N2) / 6
Answer ← 3 * (N1 - 2) + N2 / 2
IsValid ← (N1 > N2) AND (N2 = 9)
Found ← FALSE
IsValid ← (N1 > N2 / 2) OR (Found = FALSE)
Answer ← "1034" & " + " & "65"
```

(i) `Answer` ..... [1]

(ii) `Answer` ..... [1]

(iii) `IsValid` ..... [1]

(iv) `IsValid` ..... [1]

(v) `Answer` ..... [1]

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

© UCLES 2015

2 A program is to simulate the operation of a particular type of logic gate.

- The gate has two inputs (0 or 1) which are entered by the user.
- The program will display the output (0 or 1) from the gate.

The program uses the following identifiers in the pseudocode below:

Identifier	Data type	Description
P	INTEGER	Input signal
Q	INTEGER	Input signal
X	INTEGER	Output signal

```
01 INPUT P
02 INPUT Q
03 IF (P = 1 AND Q = 0) OR (P = 0 AND Q = 1) OR (P = 0 AND Q = 0)
04    THEN
05       X ← 0
06    ELSE
07       X ← 1
08 ENDIF
09 OUTPUT X
```

(a) The programmer chooses the following four test cases.

Show the output (X) for each test case.

Test case	Input		Output
	P	Q	X
1	1	1	
2	1	0	
3	0	1	
4	0	0	

[4]

(b) The selection statement (lines 03 – 08) could have been written with more simplified logic.

Rewrite this section of the algorithm in **pseudocode**.

.....

.....

.....

.....

.....

.....[3]

3 Regular customers at a supermarket use a rewards card at the point-of-sale.

Points are calculated from every transaction and added to the points total stored on the card.

One reward point is given for every $1 spent.

When the points total exceeds 500, the customer can either:

- pay the full amount due and increase their points total
- get $1 deducted from the amount due in exchange for 500 reward points

The new points total and amount to be paid is printed on the receipt.

A program is to be written with the following specification:

- read the points total from the card
- process the amount spent
- output the amount to be paid and the new points total

A user-defined function `CalculatePoints` has already been coded to calculate the new points earned from the amount spent.

Study the following pseudocode:

```
INPUT AmountDue
NewPoints ← CalculatePoints(AmountDue)
PointsTotal ← PointsTotal + NewPoints

IF PointsTotal > 500
   THEN
      OUTPUT "Exchange points?"
      INPUT Response
      IF Response = "YES"
         THEN
            PointsTotal ← PointsTotal - 500
            AmountDue ← AmountDue - 1
      ENDIF
 ENDIF

 OUTPUT AmountDue, PointsTotal
```

The algorithm is also to be documented with a program flowchart.

Complete the flowchart by:

- filling in the flowchart boxes
- labelling, where appropriate, lines of the flowchart

[6]

4 The standard pack of playing cards has four suits – called Clubs, Diamonds, Hearts and Spades. Each card has a value shown by its number or a name: 1 (Ace), 2, 3, … 10, 11 (Jack), 12 (Queen), 13 (King). The pack of cards has one combination for each suit and value.

A program is to be written which simulates a magician dealing all 52 cards from the card pack.

The program generates pairs of random numbers:

- the first, in the range 1 to 4, to represent the suit
- the second, in the range 1 to 13, to represent the card value

(a) Explain why the generation of 52 (4 suits x 13 card values) pairs of random numbers will not simulate the dealing of the complete pack.

.....

.....

.....[2]

(b) A representation of dealing out the cards is shown below:

Suit number	**Card value**												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1 (Clubs)	F	F	F	F	F	F	F	F	F	F	T	F	F
2 (Diamonds)	F	F	F	F	F	F	F	F	F	F	F	F	F
3 (Hearts)	F	F	T	F	F	F	F	F	F	F	F	F	F
4 (Spades)	F	F	F	F	F	F	F	F	F	F	F	F	F

The table shows two cards have been dealt so far; the 3 of Hearts and the Jack of Clubs.

When each card is dealt, the appropriate cell changes from F to T.

The program will output the suit and the card value in the order in which the cards are dealt.

Question 4(b) continues on page 8.

The program design in pseudocode is produced as follows:

```
DECLARE SuitNum      : INTEGER
DECLARE CardValue    : INTEGER
DECLARE DealCount    : INTEGER
DECLARE NewCard      : BOOLEAN
DECLARE CardPack ..................................................................

CALL InitialiseCardPack
DealCount ← 0
WHILE DealCount <> 52
   NewCard ← FALSE
   WHILE NewCard = FALSE
      SuitNum ← RANDOM(1,4)    // generates a random number
      CardValue ← RANDOM(1,13) // in the range given
      IF CardPack[SuitNum, CardValue] = FALSE
         THEN
            CardPack[SuitNum, CardValue] ← TRUE
            NewCard ← TRUE
            OUTPUT SuitNum, CardValue
      ENDIF
   ENDWHILE
   DealCount ← DealCount + 1
ENDWHILE

// end of main program

PROCEDURE InitialiseCardPack
   DECLARE i : INTEGER
   DECLARE j : INTEGER
   FOR i ← 1 TO 4
      FOR j ← 1 TO 13
         CardPack[i, j] ← FALSE
      ENDFOR
   ENDFOR
ENDPROCEDURE
```

Study the pseudocode and answer the questions below:

Give the line number for:

(i) A statement which marks the end of a count controlled loop.

.....[1]

(ii) The declaration of a local variable.

.....[1]

(iii) The initialisation of a variable used as a counter, but not to control a 'count controlled' loop.

.....[1]

(iv) A statement which uses a built-in function of the programming language.

.....[1]

(c) Give the number of procedures used by the pseudocode.

.....[1]

(d) Copy the condition which is used to control a 'pre-condition' loop.

.....[1]

(e) Explain the purpose of lines `14` – `19` in the design.

.....

.....

.....

.....

.....

.....[2]

(f) Complete the declaration of the global variable at line `05`.

`05 DECLARE CardPack` .....[1]

(g) Line `18` in the design shows which new card is dealt each time.

When an Ace, Jack, Queen or King is dealt, the output displays the number for that card, not the name of the card.

Card value	Card name
1	Ace
11	Jack
12	Queen
13	King

A new requirement is to display the name of the card, where appropriate.

Write a CASE structure using variable `CardValue`.

Assign to a new variable `CardName` either:

- the card value (2, 3, 4, 5, 6, 7, 8, 9 or 10)
- or where appropriate, the card name Ace, Jack, Queen or King

.....

.....

.....

.....

.....

.....

.....

.....[4]

5 A program is to process a set of integers.

The integers are stored in an array, `Num`. The first `N` elements are to be processed.

The pseudocode for this program is shown below:

```
FOR i ← 1 TO (N - 1)
   j ← 1
   REPEAT
      IF Num[j] > Num[j + 1]
         THEN
            Temp ← Num[j]
            Num[j] ← Num[j + 1]
            Num[j + 1] ← Temp
      ENDIF
      j ← j + 1
   UNTIL j = (N - i + 1)
ENDFOR
```

(a) **(i)** Trace the execution of the pseudocode for the value `N` = 5 and the given array of integers.

N	i	j	Temp		Num 1	2	3	4	5
5					11	16	13	7	8

[8]

(ii) State the purpose of the algorithm.

.....

.....[1]

(iii) Describe what evidence from the trace table suggests that the given pseudocode is inefficient.

.....

.....[1]

(b) Complete the identifier table documenting the use of each of the variables.

Identifier	Data type	Description
`Num`	`ARRAY[1:100] OF INTEGER`	The array of numbers.
`N`		
`i`		
`j`		
`Temp`		

[5]

6 Some pseudocode statements follow which use the following built-in functions:

```
ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
```
returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.
For example: `ONECHAR("Barcelona", 3)` returns 'r'.

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```
returns the number of characters in the string `ThisString`.
For example: `CHARACTERCOUNT("South Africa")` returns 12.

(a) Study the following pseudocode statements.

Give the values assigned to variables `x` and `y`.

(i) `x ← CHARACTERCOUNT("New Delhi") + 3` x ..... [1]

(ii) `y ← ONECHAR("Sri Lanka", 5)` y ..... [1]

(b) A program is to be written as follows:
- the user enters a string
- the program will form a new string with all <Space> characters removed
- the new string is output

```
NewString ← ""
INPUT InputString

j ← CHARACTERCOUNT(InputString)
FOR i ← 1 TO j
   NextChar ← ONECHAR(InputString, i)
   IF NextChar <> ""
      THEN
         // the & character joins together two strings
         NewString ← NewString & NextChar
   ENDIF
ENDFOR

OUTPUT NewString
```

(i) Complete the identifier table below.

Identifier	Data type	Description
InputString	STRING	The string value input by the user

[4]

(ii) An experienced programmer suggests this pseudocode would be best designed as a function.

Complete the re-design of the pseudocode as follows:

The main program:

- the user enters `MyString`
- the function is called and the changed string is assigned to variable `ChangedString`

The function:

- has identifier `RemoveSpaces`
- has a single parameter
- will include the declaration for any local variables used by the function

```
// main program
INPUT MyString
ChangedString←RemoveSpaces(.....................................................)
OUTPUT ChangedString

// function definition

FUNCTION RemoveSpaces(.....................................................) RETURNS .....

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

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

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

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

   j ← CHARACTERCOUNT(InputString)
   FOR i ← 1 TO j
      NextChar ← ONECHAR(InputString, i)
      IF NextChar <> " "
         THEN
            // the & character joins together two strings
            NewString ← NewString & NextChar
      ENDIF
   ENDFOR

   ..............................................................................
ENDFUNCTION
```

[7]

7 ASCII character codes are used to represent a single character.

Part of the code table is shown below.

ASCII code table (part)

Character	Decimal	Character	Decimal	Character	Decimal
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

Some pseudocode statements follow which use these built-in functions:

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```
returns the number of characters in the string `ThisString`.
For example: `CHARACTERCOUNT("South Africa")` returns 12.

```
CHR(ThisInteger : INTEGER) RETURNS CHAR
```
returns the character with ASCII value `ThisInteger`.
For example: `CHR(66)` returns 'B'.

```
ASC(ThisCharacter : CHAR) RETURNS INTEGER
```
returns the ASCII value for character `ThisCharacter`.
For example: `ASC('B')` returns 66.

(a) Give the values assigned to the variables `A`, `B`, `C` and `D`.

The & operator is used to concatenate two strings.

The expression could generate an error; if so, write ERROR.

```
Num1 ← 5
A ← ASC('F') + Num1 + ASC('Z')
B ← CHR(89) & CHR(69) & CHR(83)
C ← CHARACTERCOUNT(B & "PLEASE")
D ← ASC(ONECHAR("CURRY SAUCE", 7))
```

(i) `A` ..... [1]

(ii) `B` ..... [1]

(iii) `C` ..... [1]

(iv) `D` ..... [1]

(b) A program is to be written to input a message string and then encrypt the message.

Study the following pseudocode:

```
OUTPUT "Enter message"
INPUT MyMessage
EncryptString ← ""

FOR i ← 1 TO CHARACTERCOUNT(MyMessage)
   NextNum ← ASC(ONECHAR(MyMessage, i) + 3)
   EncryptString ← EncryptString & CHR(NextNum)
ENDFOR

OUTPUT EncryptString
```

(i) Write the above pseudocode algorithm as **program code**.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

(ii) Describe the encryption algorithm used to encrypt the message string entered by the user.

.....

.....

.....

.....[2]

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

7911807388

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2016**

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 **16** printed pages.

DC (LK/SG) 106693/4
© UCLES 2016

CAMBRIDGE International Examinations

There is an **Appendix** on the last page. Some questions will refer you to this information.

1 The items in the table below are statements from a program in a generic programming language.

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

(a) (i) Show what type of programming construct each statement represents.

Complete the table by putting a tick (✓) in the appropriate column for each item.

Item	Statement	Selection	Iteration	Assignment
1	`WHILE DegF > 37.5`			
2	`MyName = "Gordon"`			
3	`DegF = INT(DegF)`			
4	`ENDIF`			
5	`CASE OF MyFavourite`			
6	`UNTIL x = 5`			

[6]

(ii) State the purpose of each statement in the table in **part (a)(i)**.

Do **not** use mathematical symbols in your descriptions.

Item	Purpose of statement
1	....
2	....
3	....
4	....
5	....
6	....

[6]

(iii) Evaluate the following expressions when `MyString` has the value `"Corrective Maintenance"`:

Expression	Evaluates to
`'P' & MID(MyString, 13, 4)`	
`RIGHT(MID(MyString, 6, 10) ,4)`	

[2]

2 The engine management system of a car includes an energy-saving facility. When certain conditions are met, this facility will automatically stop the engine.

The system is to be software-based. It will include a subroutine, `EnergySaver`, which repeatedly takes data from sensors in the car. The subroutine decides whether or not to set the `EngineStop` value.

The table of identifiers used by this subroutine is shown below.

(a) Complete the identifier table below by stating the data types.

Identifier	Data type	Description
`Accelerator`	.....	Accelerator pedal position Values: 0 to 100 in steps of 1 Meaning: 0: none (not pressed) 100: maximum (fully pressed)
`EngineTemp`	.....	Engine temperature in °C (-50 to +150 stored to 1 decimal place)
`NormalTemp`	.....	Normal engine temperature in °C Whole number; typical value 90
`Speed`	.....	Road speed of car (in km/hr) Values: 0 to 200 in steps of 1
`EngineStop`	.....	Value used to signal engine must be stopped Possible values: TRUE: stop engine FALSE: run engine

[5]

The condition for stopping the engine is that all three of the following are true:

- Accelerator is not pressed
- Engine temperature is normal or above
- Car speed is zero

The initial design stage will produce a prototype of `EnergySaver`, with a user interface.

The structured English for this is:

1. INPUT value for accelerator pedal position
2. INPUT value for engine temperature
3. INPUT value for normal engine temperature
4. INPUT value for car speed
5. EVALUATE engine stopping condition
6. IF stopping condition satisfied SET engine stop value to TRUE
7. IF stopping condition not satisfied SET engine stop value to FALSE
8. OUTPUT message indicating engine stop value

(b) Write the **pseudocode** equivalent of the structured English. Use the identifiers from the table in **part (a)**.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

3 String encryption was implemented using a simple character-substitution method. A function, `Decrypt`, is needed to reverse the encryption process and return the original character.

The encryption uses the 7-bit ASCII value for each character. This value is used as an index for a 1D array, `Lookup`, which contains the substitute characters.
`Lookup` contains an entry for each of the ASCII characters.

This function, `Decrypt`, will accept two parameters, a single character, `CipherChar`, and the 1D array, `Lookup`.

The steps involved in `Decrypt` are follows:

- Search for the character in the array
- Note the index value where the character is found (the index value is the ASCII value of the original character)
- Use the index value to obtain the original character

(a) The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode.

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

```
FUNCTION Decrypt(................................ , ................................) RETURNS CHAR

   DECLARE Found      : ..............................................................................

   DECLARE ..............................................................................  : INTEGER

   DECLARE OriginalChar      : CHAR

   Index ← 1         //..............................................................................

   Found ← FALSE
   //search for CipherChar in Lookup:

   WHILE ..............................................................................................
      //compare CipherChar with this array element:

      IF ..............................................................................................
         THEN
            ..............................................................//Set the flag
         ELSE
            Index ..............................................//Move to next array element
      ENDIF
   ENDWHILE
   //dropped out of loop so must have found CipherChar:

   ..............................................................//convert Index to original character

   RETURN ..............................................................................................

ENDFUNCTION
```

[11]

Question 3 continues on page 8.

(b) A program is to be written to output part of the `Lookup` array.

The design of the algorithm is shown below.
It may be assumed that the characters output from `Lookup` are all printable.

For example, for the input of 65 and 3, the output will be:

```
Index 65: Character A has substitute character Y
Index 66: Character B has substitute character Q
Index 67: Character C has substitute character F
```

Write **program code** to implement the flowchart design.

In addition to the `Lookup` array, assume that the following variables have been declared:

`StartIndex, NumberToOutput, Index`

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[6]

4 **(a)** Name **two** features of your chosen high-level programming language that support the implementation of a modular design.

1 .....

2 .....
[2]

(b) **(i)** The structure chart shows part of the design of a program for an online shopping system.

The user has already added a number of products to their virtual basket.

Draw on the chart, the symbol to show that the process of modifying the basket contents may be iterated (repeated).

[1]

(ii) Each arrow in the structure chart above represents a parameter.

The table below shows the three data items that the six parameters pass between modules.

Tick (✓) to match each parameter to the correct data item.

Data item	Parameter					
	A	B	C	D	E	F
Product ID						
Quantity						
Flag Value – indicating operation success or fail						

[4]

5 Claudia stores her large collection of music CDs in different places. Claudia wants to record where she stores each CD. She decides to write a program to do this.

Data items for a typical CD are:

Title: Kind of Green
Artist: Miles Coltrane
Location: Rack3-23

The data is to be stored in a text file, `MyMusic`. Each line of the text file will be a string, formed by concatenating the three data items.

Before concatenation, the title and artist will each be made into a fixed-length string of 40 characters. Space characters may need to be added to each data item.

The location is always 8 characters long.

(a) **(i)** Explain the benefit of making the stored data into fixed-length strings.

.....

.....

.....

.....

State a drawback of this file design.

.....

.....
[3]

(ii) When Claudia buys a new CD, the CD data must be added to the existing file, `MyMusic`. She has written a procedure in pseudocode. This has the following file-handling statements:

```
OPENFILE "MyMusic" FOR WRITE
WRITEFILE "MyMusic", OutputString
CLOSEFILE "MyMusic"
```

There is a problem with the logic of this pseudocode.

State the problem.

.....

.....

Identify the effect it will have if the final code is implemented in this way.

.....

.....

Give a possible solution.

.....

.....
[3]

(b) Claudia needs to output a list of all the CDs in a particular location.

She designs a procedure, `OutputLocationList`, to do this. She also chooses the following identifiers:

Identifier	Data type
CDTitle	STRING
CDArtist	STRING
CDLocation	STRING

The procedure will:

- prompt for the name of the location
- input the location (such as “Rack3-23”)
- search the file for all CDs at this location
- output the title and artist of each CD found
- output the total number of CDs found at that location (such as “17 CDs found”)

Write **program code** for the procedure `OutputLocationList`.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[10]

6 A string-handling function has been developed.

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

The pseudocode for this function is shown below.

```
FUNCTION SF(ThisString : STRING) RETURNS STRING
   DECLARE x         : CHAR
   DECLARE NewString : STRING
   DECLARE Flag      : BOOLEAN
   DECLARE m, n      : INTEGER

   Flag ← TRUE
   NewString ← ""
   m ← LENGTH(ThisString)

   FOR n ← 1 TO m

      IF Flag = TRUE
         THEN
            x ← UCASE(MID(ThisString, n, 1))
            Flag ← FALSE
         ELSE
            x ← LCASE(MID(ThisString, n, 1))
      ENDIF

      NewString ← NewString & x

      IF x = " "
         THEN
            Flag ← TRUE
      ENDIF

   ENDFOR

   RETURN NewString
ENDFUNCTION
```

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

```
SF("big BEN")
```

n	x	Flag	m	NewString

[4]

(ii) Describe the purpose of function `SF`.

.....

.....

.....

.....[2]

(b) Test data must be designed for the function `SF`.

(i) State what happens when the function is called with an empty string.

.....

.....[1]

(ii) The function should be thoroughly tested.

Give **three** examples of non-empty strings that may be used.

In each case explain why the test string has been chosen.

String .....

Explanation .....

.....

String .....

Explanation .....

.....

String .....

Explanation .....

.....
[3]

Appendix

Built-in functions (Pseudocode)

In each function below, if the function call is not properly formed, the function returns an error.

```
MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING
```

returns the string of length `y` starting at position `x` from `ThisString`
Example: **`MID("ABCDEFGH", 2, 3)`** will return string **`"BCD"`**

```
LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING
```

returns the leftmost `x` characters from `ThisString`
Example: **`LEFT("ABCDEFGH", 3)`** will return string **`"ABC"`**

```
RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING
```

returns the rightmost `x` characters from `ThisString`
Example: **`RIGHT("ABCDEFGH", 3)`** will return string **`"FGH"`**

```
CHR(x : INTEGER) RETURNS CHAR
```

returns the character whose ASCII value is `x`
Example: **`CHR(87)`** will return **`'W'`**

```
ASC(x : CHAR) RETURNS INTEGER
```

returns the ASCII value of character `x`
Example: **`ASC('W')`** will return **`87`**

```
LCASE(x : CHAR) RETURNS CHAR
```

returns the lower case equivalent character of `x`
Example: **`LCASE('W')`** will return **`'w'`**

```
UCASE(x : CHAR) RETURNS CHAR
```

returns the upper case equivalent character of `x`
Example: **`UCASE('h')`** will return **`'H'`**

```
INT(x : REAL) RETURNS INTEGER
```

returns the integer part of `x`
Example: **`INT(27.5415)`** will return **`27`**

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2016

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

2970875529

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 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 **20** pages. Any blank pages are indicated.

DC (CJ/SG) 313087/6
© UCLES 2021

1 Part of the procedure `InitVars()` is shown:

```
PROCEDURE InitVars()
   DECLARE Var1 : INTEGER  // daily rainfall
   DECLARE Var2 : REAL     // the average wind speed
   DECLARE Var3 : STRING   // the ID number of the weather station

ENDPROCEDURE
```

(a) Give a **more meaningful** identifier name for each of the following variables used by the procedure.

Variable	New identifier name
Var1	
Var2	
Var3	

[3]

(b) A function called `ProcessVars()` assigns values to variables as shown.

Variable	Value
HouseCount	12
Turnout2018	20.23
TidalRiskCategory	'C'
IsConservationArea	FALSE
StationLocationName	"Ocean Boulevard"

Some pseudocode expressions in the function are shown in the following table.

Complete the table by evaluating each expression and writing the answer in the **Evaluates to** column.

If the expression is invalid, write “ERROR” in the **Evaluates to** column.

Refer to the **Appendix** on pages 18–19 for a list of built-in pseudocode functions and operators.

Pseudocode expression	Evaluates to
LENGTH(HouseCount) > 6	
MOD(INT(Turnout2018) * 3, 4)	
ASC(TidalRiskCategory) + Turnout2018	
IsConservationArea AND (HouseCount <= 50)	
MID(StationLocationName, 1, 5) & " Eleven"	

[5]

(c) The function header for `ProcessVars()` is:

```
FUNCTION ProcessVars(DataItem : REAL) RETURNS REAL
```

The following is an example of a parameter value passed to the function.

```
"L-MH-245,ManorHouse,N,F,230,12.34,0.98,12,N"
```

There is an error in the function header.

State the error **and** write the correct function header.

Error .....

.....

Correct function header .....

.....

[2]

(d) A programmer writes program code using an Integrated Development Environment (IDE).

Define the following **three** features of an IDE.

Breakpoints .....

.....

.....

Report (watch) window .....

.....

.....

Single stepping .....

.....

.....

[3]

2 **(a)** The following pseudocode function counts the occurrences of a character in a string.

Line numbers are shown for reference only.

```
01   DECLARE Message : STRING
02
03   FUNCTION CharacterCount(Letter : CHAR) RETURNS INTEGER
04
05      DECLARE LetterCount, Index : INTEGER
06      DECLARE ThisChar : INTEGER
07
08      LetterCount ← 1
09
10      FOR Index ← 1 TO LENGTH(Message) - 1
11          ThisChar ← LEFT(Message, Index, 1)
12          IF ThisChar = Letter
13            THEN
14              LetterCount ← LetterCount + 1
15          ENDIF
16      ENDFOR
17      RETURN LetterCount
18   ENDFUNCTION
```

(i) State the technical name of the iterative control structure used in this function.

..... [1]

(ii) Examine the pseudocode and write the answer in the table for each item.

Item	Answer
The scope of the variable `Message` is	
The start and end line numbers of a selection structure	
The identifier name of a user-defined function is	
An arithmetic operator used in the function is	

[4]

(b) Four lines of the pseudocode given in **part (a)** contain an error.

Identify the errors **and** write the correct pseudocode.

Line number for error 1 .....

Correct pseudocode .....

.....

Line number for error 2 .....

Correct pseudocode .....

.....

Line number for error 3 .....

Correct pseudocode .....

.....

Line number for error 4 .....

Correct pseudocode .....

.....

[4]

(c) A procedure, `Frequency()`, outputs the number of times each vowel occurs in a string.

The procedure will:

- prompt and input a string
- count the occurrence of each vowel in the string using a `CASE` structure
- output each vowel with its count value.

You may assume that vowels are the upper-case or lower-case characters `'a'`, `'e'`, `'i'`, `'o'`, and `'u'`.

Write **pseudocode** for the procedure `Frequency()`.

Refer to the **Appendix** on pages 18–19 for a list of built-in pseudocode functions and operators.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

3 An aeroplane carries cases.

A case is not allowed on the aeroplane if:

- the total number of cases previously checked in for a flight is 300 or more
- the weight of a case exceeds the maximum of 50 kilograms.

The following statements apply:

- Each flight is identified by a five-character string variable `FlightNo`
- Each case is identified by a three-character string variable `CaseID`
- The weight of each case is represented by a two-digit numeric string variable `CaseWeight`

A text file, `HOLD-CARGO.txt`, stores data for cases on all flights. The format of each line of the file is:

```
<FlightNo><CaseID><CaseWeight>
```

(a) A procedure, `CheckWeight()`, is required as follows:

- take the flight number as parameter
- output the list of case IDs for the flight that are not allowed on the aeroplane.

Write **pseudocode** for the procedure `CheckWeight()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(b) A decision is made to use constants in the procedure `CheckWeight()` to represent the maximum weight and the maximum number of cases.

State **two** advantages of using constants instead of variables.

Advantage 1 .....

.....

Advantage 2 .....

.....
[2]

(c) `CheckWeight()` is a user-defined function.

(i) State **two** benefits of user-defined functions.

Benefit 1 .....

.....

Benefit 2 .....

.....
[2]

(ii) State **two** benefits of built-in functions.

Do **not** give the same answers as in **part (c)(i)**.

Benefit 1 .....

.....

Benefit 2 .....

.....
[2]

(d) One method of passing a parameter is by reference.

Name **and** describe another method.

Name .....

Description .....

.....

.....

.....
[3]

4 A willow tree is a type of tree that grows at an angle. A 1D array, `TreeAngle`, has 20 elements of integer type. Each element stores the angle of a willow tree measured in degrees.

A procedure called `SafetyCheck()` is required to:

- check each tree angle in the array and count the number of trees that exceed the safe limit of 36 degrees
- determine whether the count exceeds a maintenance limit stored in the global variable `MainTrigger`, in which case a maintenance visit is required
- output a suitable message if the count is less than or equal to `MainTrigger`, otherwise output a message showing the number of times the safe limit was exceeded as follows: `"Maintain 10 trees"`.

(a) The `SafetyCheck()` procedure is written in pseudocode.

Complete the following pseudocode.

Refer to the **Appendix** on pages 18–19 for a list of built-in pseudocode functions and operators.

```
PROCEDURE SafetyCheck()
   DECLARE Count : INTEGER
   DECLARE Index : INTEGER
   CONSTANT TreeCount = 20
   ........................................ ← 0
   FOR Index ← 1 TO ........................................
     IF TreeAngle[Index] > ........................................
       THEN
         Count ← Count + 1
     ENDIF
   ENDFOR
   IF ........................................ <= MainTrigger
     THEN
       OUTPUT "Maintenance not needed"
     ELSE
       OUTPUT "Maintain " & NUM_TO_STRING(Count) & " trees"
   ENDIF
ENDPROCEDURE
```

[4]

(b) The `TreeAngle` array is changed to a 2D array that contains 20 rows and 2 columns of integer numbers as follows:

- The first column stores the integer reference number of the tree.
- The second column stores the integer angle of the tree.

Study this example:

```
TreeAngle[15, 1] ← 767 // tree reference
TreeAngle[15, 2] ← 12  // the angle of the tree
```

A procedure, `CheckTree()`, is called with an integer parameter representing the tree's reference number.

The procedure will:

- search the `TreeAngle` array for the reference number
- when a match is found, prompt and input the angle of the tree, store the data in the second dimension of the `TreeAngle` array, and then output a message if its safety status has changed
- when a match is not found, output the reference number and `"No match"`.

Write **program code** for the procedure `CheckTree()`.

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]

5 **(a)** Module iteration and module selection are features of a structure chart.

State **two other** features that may be represented in a structure chart.

Feature 1 .....

.....

Feature 2 .....

.....

[2]

(b) Examine this structure chart of a booking system.

Complete the following table by writing the correct parameter letter (A, B, C, D or E) for each parameter identifier.

Parameter identifier	Parameter letter
Quantity	
BookingID	
ItemCost	
TotalCost	
BookingDate	

[5]

6 A geocode is a string that specifies a geographical location.

The string consists of two alphanumeric characters which are followed by:

- the character `'+'`
- two more alphanumeric characters
- a comma
- a description of the location.

Here is an example of an email message that contains two geocodes. The first is for Coventry in the UK and the second is for Cambridge in the UK.

```
"This is my previous address 65+ER,CoventryUK.

I have now moved to my new location at 64+AA,CambridgeUK."
```

(a) A programmer decides to store the geocodes in a 1D array called `Location`. The array contains 10 000 elements and unused array elements are initialised with the string `"22+VV"`.

Write **pseudocode** to declare and initialise the array `Location`.

.....

.....

.....

.....

.....

.....

.....

..... [3]

(b) The following example pseudocode uses the function `FindGeoCodeIndex()` to search the array `Location` for the index position of a given geocode.

```
DECLARE CodePosition : INTEGER

CodePosition ← FindGeoCodeIndex(GeoCode)
IF CodePosition <> -1
  THEN
    OUTPUT Geocode & " found at " & NUM_TO_STRING(CodePosition)
ENDIF
```

Complete the program flowchart to represent the algorithm **only** for the function `FindGeoCodeIndex()`. Variable declarations are **not** required in program flowcharts.

[4]

(c) The function `RetrieveCode()` will extract a geocode from an email message.

Here is a summary of the design requirements for this function.

Parameters	Returns	Example
• A string containing the email message • An integer value containing the start position of the geocode in the message string	• A string containing the geocode	`Message ← "This is my previous address 65+ER,CoventryUK."` `RetrieveCode(Message, 29)` returns `"65+ER,CoventryUK"`

Assume that:

- the integer value points to the start of a valid geocode
- if a valid geocode is not at the end of the email message, it will be followed by a space or a full stop.

Write **program code** for the function `RetrieveCode()`.

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]

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 string `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"`

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

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

`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`

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

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

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

Example: `INT(27.5415)` returns `27`

```
MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10, 3)` returns `1`

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.

© UCLES 2021

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 9 6 4 4 5 1 3 5 9 3 *

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2018**

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 **16** printed pages.

DC (NH/CT) 150984/4
© UCLES 2018

CAMBRIDGE International Examinations

Question 1 begins on the next 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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2018

1 **(a)** The following table contains statements written in pseudocode.

Show what type of programming construct each statement represents.

Put a tick (✓) in the appropriate column for each statement.

Statement	Selection	Repetition (Iteration)	Assignment
`WHILE Count < 20`			
`Count ← Count + 1`			
`IF MyGrade <> 'C' THEN`			
`Mark[Count] ← GetMark(StudentID)`			
`ELSE OUTPUT "Fail"`			
`ENDFOR`			

[6]

(b) **(i)** The following table contains statements written in pseudocode.

Give the most appropriate data type for the variable used in each statement.

Statement	Data type
`MyAverage ← 13.5`	
`ProjectCompleted ← TRUE`	
`Subject ← "Home Economics"`	
`MyMark ← 270`	
`MyGrade ← 'B'`	

[5]

(ii) The following table contains statements written in pseudocode.

Complete the table by evaluating each expression using the values from **part (b)(i)**.

If any expression is invalid, write “ERROR” in the **Evaluates to** column.

For the built-in functions list, refer to the **Appendix** on page 16.

Expression	Evaluates to
`"Air-" & MID(Subject, 7, 3)`	
`INT(MyAverage / 2)`	
`ProjectCompleted AND MyMark > 270`	
`ProjectCompleted OR MyMark > 260`	
`ASC(MyGrade / 3)`	

[5]

2 Shop customers have a discount card with a unique card number. Customers collect points after they have bought items. The more points they have, the bigger the discount. If they shop on a Wednesday, their discount is increased by 20%.

The function `GetDiscountRate()` takes a card number as a parameter and returns the discount rate for a customer based on the number of points they have collected. A flowchart for the function is shown.

The function uses the following variables and functions.

Identifier	Data type	Description
`DRate`	`REAL`	The discount rate
`CardNum`	`STRING`	The unique customer card number
`Points`	`INTEGER`	The number of points collected
`GetPoints()`	`FUNCTION`	Takes the card number as a parameter and returns the number of points already collected
`Today()`	`FUNCTION`	Returns the day number: 1 for Monday, 2 for Tuesday etc.

(a) Write **pseudocode** to implement the `GetDiscountRate()` function.

Your solution should follow the flowchart for the function as closely as possible. Variable declarations should be included.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

(b) A programmer writes the function `GetDiscountRate()` in a high-level language.

(i) A run-time error could occur when the function is used.

Name **and** describe **one** other type of error that the function could contain.

Name .....

Description .....

.....

.....
[2]

(ii) Function `GetPoints()` has not been written yet.

Name **and** describe a strategy that can be used to test `GetDiscountRate()` before the `GetPoints()` function has been written.

Name .....

Description .....

.....

.....
[2]

(c) There are different ways to minimise the risk of errors when writing programs, such as the use of constants and library routines.

(i) Identify **two** values that could be replaced by constants in the function `GetDiscountRate()`.

.....

.....[1]

(ii) Write **pseudocode** to declare **one** of the constants you have given in **part (c)(i)**.

.....[2]

(iii) Explain how the use of constants helps to minimise programming errors.

.....

.....

.....

.....[2]

(iv) Give a reason why the use of library routines helps to minimise the risk of errors when writing a program.

.....

.....[1]

(v) Constants and library routines help to minimise the risk of errors.

Name another way that you can minimise the risk of errors when writing a program. Explain how this helps.

Name .....

Explanation .....

.....

.....
[2]

3 **(a)** State why a high-level language program must be translated before it can be run.

.....

.....[1]

(b) A program runs but does not give the expected output.

Describe **two** methods you could use to find the error.

Method 1 .....

.....

.....

.....

Method 2 .....

.....

.....

.....
[4]

(c) Two testing methods are black-box and white-box. A student is choosing test data for both methods.

Tick **one or more** boxes in each row to identify the testing method each statement describes.

Statement	White-box	Black-box
The student does not need to know the structure of the code.		
The student chooses data to test every possible path through the code.		
The student chooses normal, boundary and erroneous data.		
The student chooses data to test that the program meets the specification.		

[4]

Question 4 begins on the next page.

4 Part of a program written in pseudocode is shown.

```
01 DECLARE NumElements : INTEGER
…
10 FUNCTION ScanArray(SearchString : STRING) RETURNS INTEGER
11
12    DECLARE ArrayIndex : INTEGER
13    DECLARE ArrayString : STRING
14    DECLARE NumberFound : INTEGER
15
16    ArrayIndex ← 0
17    NumberFound ← 0
18
19    FOR ArrayIndex ← 1 TO NumElements
20       ArrayString ← ResultArray[ArrayIndex, 1]
21       IF ArrayString = SearchString
22          THEN
23             CALL SaveToFile(ArrayString)
24             NumberFound ← NumberFound + 1
25       ENDIF
26    ENDFOR
27
28    RETURN NumberFound
29
30 ENDFUNCTION
```

(a) **(i)** Examine the pseudocode **and** complete the following table.

	Answer
The identifier name of a global integer	
The identifier name of a user-defined procedure	
The line number of an unnecessary statement	
The scope of `ArrayString`	

[4]

(ii) Describe in detail the purpose of lines `19` to `26` in the function `ScanArray()`.
Do **not** use pseudocode in your answer.

.....

.....

.....

.....

.....

.....

.....[4]

(b) The function `ScanArray()` needs to be amended so that the comparison is not case sensitive. For example, comparing "`Aaaa`" with "`AAAa`" should evaluate to `TRUE`.

Write **program code** to implement the **amended** `ScanArray()` function.

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) The function `ScanArray()` is one of a number of sub-tasks within a program.

Name the process that involves the splitting of a problem into sub-tasks **and** state **two** advantages of this approach.

Name .....

Advantage 1 .....

.....

Advantage 2 .....

.....
[3]

(d) `ResultArray` is a 2D array of type `STRING`. It represents a table containing 100 rows and 2 columns.

Write **program code** to declare `ResultArray` **and** set all elements to the value `'*'`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....[3]

Question 5 begins on the next page.

5 A program collects data about the performance of a car at regular time intervals. A text file, `CarStatus.txt`, stores the data.

The format of each line of the text file is as follows:

```
<Time>,<Amount of fuel used>,<Distance travelled>
```

Data items are separated by a '`,`' (comma) character.

The program contains the following functions.

Function	Description
`GetTime()`	Returns a string representing the current time. May return `NULL` under certain circumstances.
`GetFuel()`	Returns a string representing the amount of fuel used
`GetDistance()`	Returns a string representing the distance travelled

The function `SaveStatus()` will:

- obtain the time, fuel used and distance data using the appropriate function calls
- check that the time string is not `NULL`
- return `FALSE` if the current time string remains `NULL` after three attempts
- form the text string, write it to the file and return `TRUE`

The file should not be open longer than necessary.

Write **pseudocode** for the `SaveStatus()` function.

.....[10]

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

`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 string `"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 string `"ABC"`

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

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

`TO_UPPER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all lower case alphabetic characters of `ThisString` to upper case. Other characters will be unchanged.

Example: `TO_UPPER("Disk Error 27")` returns `"DISK ERROR 27"`

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

Example: `INT(27.5415)` returns `27`

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

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

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`

© UCLES 2018

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

8955017847

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2018**

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 **15** printed pages and **1** blank page.

DC (CE/SG) 146142/3
© UCLES 2018

CAMBRIDGE International Examinations

1 **(a)** A program controls the heating system of an energy-efficient house.

Give a suitable **identifier name** for each of the data items.

Description of data item	Suitable identifier name
The temperature inside the house	
The temperature outside the house	
The wind speed	
Whether it was raining or not	

[4]

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

Variable	Value
`Quality`	`'D'`
`DayNumber`	`20`
`MyName`	`"Stephen"`
`QualityConfirmed`	`TRUE`
`Factor`	`6.5`

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 15.

Expression	Evaluates to
`MID(MyName, 4, 4) & "ol"`	
`QualityConfirmed AND (Factor >= 6.5)`	
`20 + ASC(Quality)`	
`QualityConfirmed + 3`	
`MOD(Factor * 2, 9)`	

[5]

(ii) Programming languages support different data types.

Give an appropriate data type for each of these variables from **part (b)(i)**.

Variable	Data type
`QualityConfirmed`	
`DayNumber`	
`Factor`	
`Quality`	
`MyName`	

[5]

2 The following is a function design in pseudocode.

Line numbers are given for reference only.

```
01  FUNCTION GradeCalc(Mark : ARRAY[] OF INTEGER) RETURNS INTEGER
02
03    DECLARE DGradeCount : INTEGER
04    DECLARE n : INTEGER
05    DECLARE Grade : STRING
06    DECLARE ThisMark : INTEGER
07
08    DGradeCount ← 0    // initialise variables
09    n ← 1
10    ThisMark ← 1
11    REPEAT
12       ThisMark ← Mark[n]    // get next mark from Mark array
13
14       IF ThisMark > 74
15          THEN
16             Grade ← "Distinction"
17             DGradeCount ← DGradeCount + 1 // one more Distinction
18          ELSE
19             IF ThisMark > 59
20                THEN
21                   Grade ← "Merit"
22                ELSE
23                   IF ThisMark > 39
24                      THEN
25                         Grade ← "Pass"
26                      ELSE
27                         Grade ← "Fail"
28                   ENDIF
29             ENDIF
30       ENDIF
31
32       OUTPUT "Grade for Student " & n & " is " & Grade
33
34       n ← n + 1
35    UNTIL n = 101 // no more elements in the array
36
37    RETURN Grade // return the number of Distinction grades
38
39  ENDFUNCTION
40
41  DECLARE Mark : ARRAY[1:100] OF INTEGER
42  DECLARE DistinctionGrades : INTEGER
43
44  CALL GetMarks(Mark[])  // fill the array with student marks
45  DistinctionGrades ← GradeCalc(Mark[])
```

(a) This pseudocode includes features that are examples of good practice.

Explain why it is useful to include the two following features.

Comments .....

.....

Indentation .....

.....
[2]

(b) Study the function `GradeCalc()`. Identify the features of the function in the following table.

Feature	Answer
A line number containing an example of an integer assignment statement	
A line number containing the start of a selection structure	
A line number containing the end of a selection structure	
The upper bound of the `Mark` array	
The number of dimensions of the `Mark` array	
The name for the type of loop structure used	
A line number containing an unnecessary assignment statement	
The number of times `OUTPUT` is called	
The number of local variables	

[9]

(c) **(i)** There is a mistake in the pseudocode that would produce a data type mismatch error if a programmer were to write similar program code.

Describe this mistake and how it may be corrected.

.....

.....

.....

.....[2]

(ii) Lines 14 to 30 of the pseudocode on page 4 could be written using a `CASE` structure.

Write the equivalent `CASE` structure in **pseudocode**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

3 A modular program design consists of four modules:

`Module1` has three sub-tasks. Each sub-task is implemented by a single subroutine (either a function or a procedure).

The subroutine headings are defined as follows:

```
FUNCTION  Module2 (Weight : REAL) RETURNS BOOLEAN
PROCEDURE Module3 (Weight : REAL, Customer : STRING, Purchased : DATE)
FUNCTION  Module4 (Purchased : DATE, Account : INTEGER) RETURNS INTEGER
```

(a) State the term given to values passed between modules.

.....[1]

(b) Draw a structure chart to represent the program design.

Use the letters in the table to label the values passed between modules.

Value	Label
Boolean return value	A
Integer return value	B
`Account`	C
`Customer`	D
`Purchased`	E
`Weight`	F

Structure chart

[4]

4 A program controls a chemical process in a factory.

The temperature is monitored as part of the control process. The temperature is measured at fixed time intervals and the value is stored in an array, `PTemp`. The array contains 100 elements, representing 100 temperature values. The first element is `PTemp[1]`.

The program will check whether the temperature is outside the acceptable range more than 20 times. This task is performed by a function, `IsTempOK()`.

The algorithm for the function `IsTempOK()` is expressed in structured English as follows:

1 Examine each array element and count the number of times that a temperature is less than `MinTemp` or more than `MaxTemp`.
2 If the count in step 1 exceeds 20, return `FALSE`, otherwise return `TRUE`.

Draw a program flowchart, on the next page, to represent the algorithm for the function `IsTempOK()`.

Assume:

- the array contains 100 valid temperature values
- `PTemp, MinTemp` and `MaxTemp` are global variables.

Note that variable declarations are not required in program flowcharts.

[9]

5 A golf club holds information about its members. When a member completes a round of golf, their score is stored along with their membership number and the date of the round.

A program is to be written to store and process the score information.

The information to be stored is as follows:

- `MembershipNumber` is a four-digit numeric string.
- `Date` is a six-digit numeric string in the format DDMMYY
- `Score` is a two-digit numeric string in the range "`50`" to "`99`".

This information is stored in a single string in the format:

```
<MembershipNumber><Date><Score>
```

(a) **(i)** The program designer considers storing the strings in an array.

State how many dimensions the array would require.

.....[1]

(ii) The designer has decided to store the strings in a file.

Give a reason for this choice.

.....

.....[1]

(b) The program will be developed using an Integrated Development Environment (IDE). One feature provided by an IDE is known as prettyprint.

Name **two** presentation features provided by prettyprint.

Feature 1 .....

.....

Feature 2 .....

.....
[2]

(c) The strings are stored in a text file, `ScoreDetails.txt`.

The program needs a function, `GetAverageScore()`.

The structured English representing the algorithm for this function is as follows:

1 Receive a membership number as its argument.
2 Search the `ScoreDetails.txt` file for all scores for that member.
3 Calculate the average score.
4 Return the average score.

You can assume that there will be at least one score relating to each member.

Write **program code** for the `GetAverageScore()` function.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[10]

6 **(a)** Individual elements in a 1D array are referenced using an integer variable.

For example, using the variable `n`, an individual element would be referenced in pseudocode as:

```
StudentGrade[n]
```

Give the correct technical term for the variable `n`.

.....[1]

(b) A 2D array, `Picture`, contains data representing a bitmap image. Each element of the array represents one pixel of the image. The image is grey-scale encoded where the value of each pixel ranges from 0 (representing black) to 255 (representing white) with intermediate values representing different levels of grey.

The following is an example of an image and the corresponding data values for the `Picture` array.

Bitmap image

Values

240	10	10	10	10	10	10	240
80	80	240	80	80	240	80	80
80	80	240	80	80	240	80	80
80	80	150	150	150	150	80	80
80	80	240	240	240	240	80	80
80	80	150	150	150	150	80	80
240	240	150	150	150	150	240	240
240	240	150	150	150	150	240	240

In pseudocode, the array is declared as follows:

```
DECLARE Picture : ARRAY[1:8, 1:8] OF INTEGER
```

A function, `Clip()`, is required that will:

- take an integer parameter, `MaxVal`, to represent the maximum allowed value for each pixel
- limit each pixel value to the maximum allowed value
- return `TRUE` if any of the pixel values have been changed, otherwise return `FALSE`

An example use of the function is:

```
IsClipped ← Clip(234)
```

This will limit the maximum pixel value within the array to 234. Pixels with a value not greater than 234 will remain unchanged.

Write **pseudocode** to implement the `Clip()` function.

Assume that `Picture` is a global variable.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[9]

7 A function, `IsFactor()` takes two integer parameters and checks whether the second parameter is a factor of the first parameter.

For example `IsFactor(6,3)` would return `TRUE` as six divided by three has a remainder of zero.

This function returns `FALSE` if there is a non-zero remainder, or if the value of the second parameter is zero.

Write **program code** to implement the `IsFactor()` function.

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]

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 string `"BCD"`

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

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

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

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

Example: `INT(27.5415)` returns `27`

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

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

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` produces `"Summer Pudding"`
`AND`	Performs a logical `AND` of two Boolean values. Example: `TRUE AND FALSE` produces `FALSE`
`OR`	Performs a logical `OR` of two Boolean values. Example: `TRUE OR FALSE` produces `TRUE`

© UCLES 2018

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2018

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 4 5 3 1 9 9 0 1 8 1 *

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2015**

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 **19** printed pages and **1** blank page.

DC (GB) 116769
© UCLES 2015

CAMBRIDGE International Examinations

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

1 Computer programs have to evaluate expressions.

Study the sequence of pseudocode statements.

Write down the value assigned to each variable.

```
DECLARE h, w, r, Perimeter, Area : REAL
DECLARE A, B, C, D, E            : BOOLEAN
```

```
h ← 13.6
w ← 6.4
Perimeter ← (h + w) * 2
```

(i) Perimeter ..... [1]

```
r ← 10
Area 3.142 * r^2
```

(ii) Area ..... [1]

```
Z ← 11 + r / 5 + 3
```

(iii) Z ..... [1]

```
A ← NOT(r > 10)
```

(iv) A ..... [1]

2 A programmer uses an Integrated Development Environment (IDE) for all program development.

(i) Describe what is meant by an IDE.

.....

.....

.....

.....[2]

(ii) Name **three** features you would expect to be available in an IDE to help initial error detection or debugging.

1 .....

.....

2 .....

.....

3 .....

.....[3]

3 A program is to simulate the operation of a particular type of logic gate.

- The gate has two inputs (TRUE or FALSE) which are entered by the user.
- The program will display the output (TRUE or FALSE) from the gate.

The program uses the following identifiers in the pseudocode below:

Identifier	Data type	Description
`InA`	`BOOLEAN`	Input signal
`InB`	`BOOLEAN`	Input signal
`OutZ`	`BOOLEAN`	Output signal

```
INPUT InA
INPUT InB
IF (InA = FALSE AND InB = FALSE) OR (InA = FALSE AND InB = TRUE)
                                    OR (InA = TRUE AND InB = FALSE)
   THEN
      OutZ ← TRUE
   ELSE
      OutZ ← FALSE
ENDIF
OUTPUT OutZ
```

(a) The programmer chooses the following four test cases.

Show the output (`OutZ`) expected for each test case.

Test case	Input		Output `OutZ`
	`InA`	`InB`	
1	`TRUE`	`TRUE`	
2	`TRUE`	`FALSE`	
3	`FALSE`	`TRUE`	
4	`FALSE`	`FALSE`	

[4]

(b) The selection statement (lines `03` – `08`) could have been written with more simplified logic.

Rewrite this section of the algorithm in **pseudocode**.

.....

.....

.....

.....

.....

.....[3]

4 A program is to be written to calculate the discount given on purchases.

A purchase may qualify for a discount depending on the amount spent. The purchase price (`Purchase`), the discount rate (`DiscountRate`) and amount paid (`Paid`) is calculated as shown in the following pseudocode algorithm.

```
INPUT Purchase

IF Purchase > 1000
   THEN
      DiscountRate ← 0.10
   ELSE
      IF Purchase > 500
         THEN
            DiscountRate ← 0.05
         ELSE
            DiscountRate ← 0
      ENDIF
ENDIF

Paid ← Purchase * (1 - DiscountRate)
OUTPUT Paid
```

The algorithm is also to be documented with a program flowchart.

Complete the flowchart by:

- filling in the flowchart boxes
- labelling, where appropriate, lines of the flowchart

[6]

5 A driver buys a new car.

The value of the car reduces each year by a percentage of its current value.

The percentage reduction is:

- in the first year, 40%
- in each following year, 20%

The driver writes a program to predict the value of the car in future years.

The program requirements are:

- enter the cost of the new car (to nearest $)
- calculate and output the value of the car at the end of each year
- the program will end when either the car is nine years old, or when the value is less than $1000

(a) Study the incomplete pseudocode which follows in **part (b)** and fill in the identifier table.

Identifier	Data type	Description

[3]

(b) Complete the pseudocode for this design.

```
OUTPUT "Enter purchase price"
INPUT PurchasePrice

CurrentValue ← ..............................................................
YearCount ← 1

WHILE ..................................... AND ..............................

   IF ..........................................................................
      THEN
         CurrentValue ← CurrentValue * (1 - 40 / 100)
      ELSE
         CurrentValue ← ...........................................................
   ENDIF

   OUTPUT YearCount, CurrentValue

   ..............................................................................
ENDWHILE
```

[6]

6 A firm employs five staff who take part in a training programme. Each member of staff must complete a set of twelve tasks which can be taken in any order. When a member of staff successfully completes a task, this is recorded.

A program is to be produced to record the completion of tasks for the five members of staff.

To test the code, the programmer makes the program generate test data.

The program generates pairs of random numbers:

- the first, in the range, 1 to 5 to represent the member of staff
- the second, in the range, 1 to 12 to represent the task

Each pair of numbers simulates the completion of one task by one member of staff.

(a) Explain why the generation of 60 (5 staff x 12 tasks) pairs of random numbers will not simulate all tasks completed by all staff.

.....

.....

.....[2]

(b) Data is currently recorded manually as shown.

Staff number	**Task number**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1												
2												
3				✓								
4												
5								✓				

The table shows that two members of staff have each successfully completed one task.

The program must use a suitable data structure to store, for all staff:

- tasks successfully completed
- tasks not yet successfully completed

The program will output the staff number and task number in the order in which tasks are completed.

The program design in pseudocode is produced as follows:

```
DECLARE StaffNum          : INTEGER
DECLARE TaskNum           : INTEGER
DECLARE ..................................................................................
DECLARE NewStaffTask      : BOOLEAN

CALL InitialiseTaskGrid
Completed ← 0
WHILE Completed <> 60
   NewStaffTask ← FALSE
   WHILE NewStaffTask = FALSE
      StaffNum ← RANDOM(1,5)       //generates a random number
      TaskNum ← RANDOM(1,12)       //in the given range
      IF TaskGrid[StaffNum, TaskNum] = FALSE
         THEN
            TaskGrid[StaffNum, TaskNum] ← TRUE
            NewStaffTask ← TRUE
            OUTPUT StaffNum, TaskNum
      ENDIF
   ENDWHILE
   Completed ← Completed + 1
ENDWHILE
OUTPUT "Staff Task Count", Completed

// end of main program

PROCEDURE InitialiseTaskGrid()
   DECLARE i : INTEGER
   DECLARE j : INTEGER
   FOR i ← 1 TO 5
      FOR j ← 1 TO 12
         TaskGrid[i, j] ← FALSE
      ENDFOR
   ENDFOR
ENDPROCEDURE
```

Study the pseudocode and answer the questions below.

Give the line number for:

(i) The declaration of a `BOOLEAN` global variable. ..... [1]

(ii) The declaration of a local variable. ..... [1]

(iii) The incrementing of a variable used as a counter, but not to control a 'count controlled' loop.

..... [1]

(iv) A statement which uses a built-in function of the programming language.

..... [1]

(c) (i) State the number of parameters of the `InitialiseTaskGrid` procedure.

..... [1]

(ii) Copy the condition which is used to control a 'pre-condition' loop.

.....[1]

(iii) Explain the purpose of lines `13` – `18`.

.....

.....

.....

.....

.....

.....[3]

(iv) Give the global variable that needs to be declared at line `03`.

.....[2]

(d) Line `17` in the pseudocode outputs the staff number and the task number.

A new requirement is to display the name of the member of staff given in the table.

Staff number	Staff name
1	Sadiq
2	Smith
3	Ho
4	Azmah
5	Papadopoulos

Write a CASE structure using variable `StaffNum`.

Assign to a new variable `StaffName` the appropriate staff name.

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

Question 7 begins on page 14.

7 ASCII character codes are used to represent a single character.

Part of the code table is shown below.

ASCII code table (part)

Character	Decimal	Character	Decimal	Character	Decimal
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

Some pseudocode statements follow which use the built-in functions below:

`ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR`
returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.
For example: `ONECHAR("Barcelona", 3)` returns 'r'.

`CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER`
returns the number of characters in the string `ThisString`.
For example: `CHARACTERCOUNT("BRAZIL")` returns 6.

`CHR(ThisInteger : INTEGER) RETURNS CHAR`
returns the character with ASCII code `ThisInteger`.
For example: `CHR(65)` returns character 'A'.

`ASC(ThisCharacter : CHAR) RETURNS INTEGER`
returns the ASCII value for character `ThisCharacter`.
For example: `ASC('A')` returns 65.

(a) Show the values stored by variables `A`, `B`, `C` and `D`.

The & operator is used to concatenate two strings.

```
Num1 ← 15
A ← CHR(67) & CHR(65) & CHR(84)
B ← ASC('P') - ASC('F') + 3
C ← ASC(ONECHAR("BISCUITS", 3))
D ← CHARACTERCOUNT("New York City") + 2
```

(i) A ..... [1]

(ii) B ..... [1]

(iii) C ..... [1]

(iv) D ..... [1]

(b) A program is to be written which accepts a string and then calculates a numeric value from this string. The input string and the calculated value are then to be sent to a remote computer over a communications link.

Study the following pseudocode:

```
OUTPUT "Enter string"
INPUT MyString
StringTotal ← 0

FOR i ← 1 TO CHARACTERCOUNT(MyString)
   NextNum ← ASC(ONECHAR(MyString, i))
   StringTotal ← StringTotal + NextNum
ENDFOR

OUTPUT MyString, StringTotal
```

Write the above pseudocode algorithm as **program code**.

There is no need to show the declaration of variables or comment statements.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

(c) Explain the purpose of sending the value of `StringTotal` to the remote computer, in addition to `MyString`.

.....

.....

.....

.....[2]

8 In this question you will need to use the given pseudocode built-in function:

```
ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
```
returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.
For example: `ONECHAR("Barcelona", 3)` returns 'r'.

(a) Give the value assigned to variable `y` by the following statement:

`y ← ONECHAR("San Francisco", 6)` `y` ..... [1]

A program reads a string entered by the user. The string represents the addition or subtraction of two fractions. Each part of the fraction within the string is always a single digit only and the top digit is always less than the bottom digit.

Example strings are: `"3/8+3/5"` and `"5/8-1/4"`

The program steps are:
- the user enters the string
- the program isolates each digit and the operator
- the program computes the answer as either:
  - a fraction
  - a whole number followed by a fraction
  - a whole number
- the program displays the answer to the user

The identifier table shows the variables to be used to store the characters in the string as shown in the diagram.

Identifier	Data type	Description
`FractionString`	`STRING`	String input by user. For example: `"5/8-1/4"`
`N1Char`	`CHAR`	See diagram
`N2Char`	`CHAR`	See diagram
`N3Char`	`CHAR`	See diagram
`N4Char`	`CHAR`	See diagram
`Op`	`CHAR`	See diagram

(b) Study the sequence of pseudocode statements.

Show the values assigned to each variable.

```
FractionString ← "3/7+2/9"
N3Char ← ONECHAR(FractionString, 5)
Op ← ONECHAR(FractionString, 4)
```

(i) `N3Char` ..... [1]

(ii) `Op` ..... [1]

(iii) Complete the function call to isolate the character `'9'` from `FractionString`.

```
FractionString ← "3/7+2/9"
ONECHAR(FractionString, ..............)
```

[1]

The following additional variables are to be used by the program:

Identifier	Data type	Description
N1	INTEGER	The number value of N1Char
N2	INTEGER	The number value of N2Char
N3	INTEGER	The number value of N3Char
N4	INTEGER	The number value of N4Char
TopAnswer	INTEGER	The numerator of the fraction answer
BottomAnswer	INTEGER	The denominator of the fraction answer

(c) The following pseudocode uses these additional built-in functions:

`TONUM(ThisDigit : CHAR) RETURNS INTEGER` returns the integer value of character `ThisDigit` For example: `TONUM('8')` returns digit `8`.

`TOSTR(ThisNumber : INTEGER) RETURNS STRING` returns the string value of integer `ThisNumber` For example: `TOSTR(27)` returns `"27"`.

Study the pseudocode.

Complete the **three** dry runs for the three given values of `FractionString`.

```
OUTPUT "Enter the expression"
INPUT FractionString

// isolate each number digit and assign its number value
N1Char ← ONECHAR(FractionString, 1)
N1 ← TONUM(N1Char)
N2Char ← ONECHAR(FractionString, 3)
N2 ← TONUM(N2Char)
N3Char ← ONECHAR(FractionString, 5)
N3 ← TONUM(N3Char)
N4Char ← ONECHAR(FractionString, 7)
N4 ← TONUM(N4Char)

BottomAnswer ← N2 * N4

Op ← ONECHAR(FractionString, 4)
IF Op = '+'
   THEN
      // add fractions
      TopAnswer ← (BottomAnswer/N2) * N1 + (BottomAnswer/N4) * N3
   ELSE
      // subtract fractions
      TopAnswer ← (BottomAnswer/N2) * N1 - (BottomAnswer/N4) * N3
ENDIF

IF TopAnswer = BottomAnswer
   THEN
      OUTPUT '1'
   ELSE
      IF TopAnswer > BottomAnswer
         THEN
            TopAnswer ← TopAnswer MOD BottomAnswer
            // the & operator joins strings or character values
            OUTPUT "1 " & TOSTR(TopAnswer) & "/" & TOSTR(BottomAnswer)
         ELSE
            OUTPUT TOSTR(TopAnswer) & "/" & TOSTR(BottomAnswer)
      ENDIF
ENDIF
```

(i) `FractionString ← "2/5-3/8"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[2]

(ii) `FractionString ← "3/4+1/4"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[2]

(iii) `FractionString ← "7/9+2/3"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[3]

(d) The programmer writes code from the given pseudocode design. The program works, but the design is limited.

The programmer is to make amendments to the design following suggested specification changes.

(i) State the name for this type of maintenance.

.....[1]

(ii) Describe **three** specification changes which will make the program more useful.

1 .....

.....

2 .....

.....

3 .....

.....[3]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2015

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2021**

INSERT **2 hours**

2390708169–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (PQ) 222283
© UCLES 2021

Note: An error occurs if a function call is not properly formed, or if the parameters are incorrect.

STRING Functions

`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"`

`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`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the lower case equivalent of `ThisChar`
Alphabetic characters that are not upper case are unchanged.

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

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the upper case equivalent of `ThisChar`
Alphabetic characters that are not lower case are unchanged.

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

`TO_UPPER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to upper case.

Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`

`TO_LOWER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to lower case.

Example: `TO_LOWER("JIM 803")` returns `"jim 803"`

© UCLES 2021

```
NUM_TO_STR(x : <data type1>) RETURNS <data type2>
```
returns a string representation of a numeric value.

Note: `<data type1>` may be `REAL` or `INTEGER`
`<data type2>` may be `CHAR` or `STRING`

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

```
STR_TO_NUM(x : <data type1>) RETURNS <data type2>
```
returns a numeric representation of a string.

Note: `<data type1>` may be `CHAR` or `STRING`
`<data type2>` may be `REAL` or `INTEGER`

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

```
IS_NUM(ThisString : STRING) RETURNS BOOLEAN
```
returns the value `TRUE` if `ThisString` represents a valid numeric value.

Example 1: `IS_NUM("12.36")` returns `TRUE`
Example 2: `IS_NUM("-12.36")` returns `TRUE`
Example 3: `IS_NUM("12.3a")` returns `FALSE`

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

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

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

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

NUMERIC Functions

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

Example: `INT(27.5415)` returns `27`

```
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`

OTHER Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: The function will generate an error if the file is not already open in `READ` mode.

Note: An error occurs if an operator with a value of an incorrect type is used.

OPERATORS

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to "Summer Pudding"
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

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.

© UCLES 2021

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

5526189778

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2017**

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 **14** printed pages and **2** blank pages.

DC (RW/JG) 129910/3
© UCLES 2017

CAMBRIDGE International Examinations

1 **(a)** Simple algorithms usually consist of three different stages.

Complete the following table.
Add a description of the stage and an example pseudocode statement.

The first stage has been given.

Stage	Description and example
Input	Description: .... Pseudocode example: ....
.....	Description: .... Pseudocode example: ....
.....	Description: .... Pseudocode example: ....

[7]

(b) **(i)** `AND` and `OR` are two operators that may be used when implementing an algorithm.
An example of their use is given in the following pseudocode statement:

```
MyFlag ← VarA OR VarB
```

State the data type of variable `MyFlag`.

.....[1]

(ii) State the name given to the type of operators to which `AND` and `OR` belong.

.....[1]

(iii) Evaluate the expressions given in the following table when the variable values are as follows:

```
FlagA ← TRUE
FlagB ← FALSE
FlagC ← TRUE
```

Expression	Evaluates to
`FlagA AND (FlagB OR FlagC)`	
`FlagA AND (FlagB AND FlagC)`	
`(NOT FlagA) OR (NOT FlagC)`	

[3]

(c) A common construct found in many algorithms is a loop.

Using **pseudocode**, write a pre-condition loop to output all of the even numbers between 99 and 201.

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

2 One of the security features of a multi-user computer system is a user login process. The user must complete this successfully before they can access the resources of the system.

As part of the login process the user enters their user ID followed by a password. The system then compares the password entered with the password held in a file.

(a) The steps involved in the login process are described as follows:

- User enters their ID and password.
- Validation checks:
  - Compare user ID with data from the file.
  - Indicate whether or not the user ID was found.
  - If user ID found, check whether passwords match.

The description above is not detailed enough to allow a program to be written. The validation checks must be expressed as a more detailed algorithm.

Give the name of the process of increasing the level of detail of the algorithm.

.....[1]

(b) An identifier table is created as the algorithm is developed. A section of the table is shown. Complete the table.

Identifier	Data Type	Description
`UserIDInput`	.....	Stores the user ID entered
`PasswordInput`	.....	....
`UserIDFound`	.....	....
`PasswordValid`	.....	....

[5]

(c) The validation checks described in **part (a)** are to be used as the basis for program code.

Use **structured English** to write a more detailed algorithm.

You should:

- use the identifiers given in the previous table
- assume that the password file, `password.txt`, is organised as a simple text file. The user ID and password are stored together, one entry per line.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

3 A string conversion function, `StringClean`, is to be written.

This function will form a new string, `OutString`, from a given string, `InString`, by:

- removing all non-alphabetic characters
- converting all alphabetic characters to lower case.

For example:

```
InString = "Good Morning, Dave"
OutString = "goodmorningdave"
```

The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode using relevant built-in functions.

For the built-in functions list, refer to the **Appendix** on page 14.

```
FUNCTION StringClean(..........................) RETURNS ..........

   DECLARE NextChar : ....................................

   DECLARE ............................................... : STRING

   ..................................... //initialise the return string

   //loop through InString to produce OutString

   FOR n ← 1 TO ............................. //from first to last

      NextChar ← ............................ //get next character and

      NextChar ← ............................ //convert to lower case

      IF ..................................... //check if alphabetic

         THEN

            ...................................... //add to OutString

      ENDIF

   ENDFOR

   ..........................................//return value

ENDFUNCTION
```

[11]

4 **(a)** A structure chart is a tool used in modular program design.

State **three** pieces of information that a structure chart can convey about a program design.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(b) The following diagram shows part of a structure chart.

Examples of the data items that correspond to the arrows are given in this table:

Arrow	Data item
A	`234.56`
B	`"Mr Robert Zimmerman"`
C	`True`

Use **pseudocode** to write the function header for the **Card payment** module.

.....

.....[3]

5 A multi-user computer system records user login information in a text file, `LoginFile.txt`. Each time a user successfully logs into the system, the following information is recorded:

Item	Information	Example data
1	A five character user ID	`"JimAA"`
2	A four character port ID	`"3456"`
3	A fourteen character time and date	`"08:30Jun012015"`

The data items are concatenated to form a single string. Each string is saved as a separate line in the text file.

The example data in the preceding table would result in the following text line in the file:

"JimAA345608:30Jun012015"

The computer system can produce a list of the successful login attempts by a given user.

The file `LoginFile.txt` is searched for a given user ID and the corresponding data are copied into a 2D array, `LoginEvents`.

`LoginEvents` has been declared in pseudocode as:

```
DECLARE LoginEvents[1 : 1000, 1 : 2] OF STRING
```

A procedure, `SearchFile`, is needed to search the file and copy selected data to the array.

The main steps of the procedure are as follows:

- Input a user ID.
- Search `LoginFile.txt` for entries with matching user ID.
- For matching entries, copy items 2 and 3 above into the `LoginEvents` array.

You can assume that:

- the system initialises all elements of `LoginEvents` to an empty string " ", before it calls `SearchFile`
- there will be no more than 1000 successful logins for a single user.

Write **program code** for the procedure `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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[10]

6 A multi-user computer system makes use of passwords.

To be valid, a password must comply with the following rules:

- at least two lower-case alphabetic characters
- at least two upper-case alphabetic characters
- at least three numeric characters
- alpha-numeric characters only

A function, `ValidatePassword`, is needed to check that a given password follows these rules. This function takes a string, `Pass`, as a parameter and returns a Boolean value:

- `TRUE` if `Pass` contains a valid password
- `FALSE` otherwise.

(a) Write **program code** to implement the new function `ValidatePassword`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[10]

(b) **(i)** The function will be tested.

Give a valid string to check that the function returns `TRUE` under the correct conditions.

String1: .....

Modify the valid string given for String1 to test each rule separately.

Explain your choice in each case.

String2: .....

Explanation: .....

.....

.....

String3: .....

Explanation: .....

.....

.....

String4: .....

Explanation: .....

.....

.....

String5: .....

Explanation: .....

.....

.....
[5]

(ii) When testing a module, it is necessary to test all possible paths through the code.

State the name given to this type of testing.

.....[1]

(iii) A program consisting of several modules may be tested using a process known as stub testing.

Explain this process.

.....

.....

.....

.....[2]

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

```
MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING
```

returns string of length `y` starting at position `x` from `ThisString`.
Example: **`MID("ABCDEFGH", 2, 3)`** returns string **`"BCD"`**

```
LENGTH(ThisString : STRING) RETURNS INTEGER
```

returns the integer value representing the length of string `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 string **`"ABC"`**

```
RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING
```

returns rightmost `x` characters from `ThisString`.
Example: **`RIGHT("ABCDEFGH", 3)`** returns string **`"FGH"`**

```
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 then it is returned unchanged.
Example: **`LCASE('W')`** returns **`'w'`**

```
MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```

returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`.
Example: **`MOD(10,3)`** returns **`1`**

```
DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```

returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`.
Example: **`DIV(10,3)`** returns **`3`**

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings. Example: **`"Summer" & " " & "Pudding"`** produces **`"Summer Pudding"`**
`AND`	Performs a logical **`AND`** of two Boolean values. Example: **`TRUE AND FALSE`** produces **`FALSE`**
`OR`	Performs a logical **`OR`** of two Boolean values. Example: **`TRUE OR FALSE`** produces **`TRUE`**

© UCLES 2017

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2017

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

8270163620

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2017**

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 **14** printed pages and **2** blank pages.

DC (ST/JG) 144340
© UCLES 2017

CAMBRIDGE International Examinations

1 **(a)** Simple algorithms usually consist of three different stages.

Complete the following table.
Add a description of the stage and an example pseudocode statement.

The first stage has been given.

Stage	Description and example
Input	Description: .... Pseudocode example: ....
.....	Description: .... Pseudocode example: ....
.....	Description: .... Pseudocode example: ....

[7]

(b) **(i)** `AND` and `OR` are two operators that may be used when implementing an algorithm.
An example of their use is given in the following pseudocode statement:

```
MyFlag ← VarA OR VarB
```

State the data type of variable `MyFlag`.

.....[1]

(ii) State the name given to the type of operators to which `AND` and `OR` belong.

.....[1]

(iii) Evaluate the expressions given in the following table when the variable values are as follows:

```
FlagA ← TRUE
FlagB ← FALSE
FlagC ← TRUE
```

Expression	Evaluates to
`FlagA AND (FlagB OR FlagC)`	
`FlagA AND (FlagB AND FlagC)`	
`(NOT FlagA) OR (NOT FlagC)`	

[3]

(c) A common construct found in many algorithms is a loop.

Using **pseudocode**, write a pre-condition loop to output all of the even numbers between 99 and 201.

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

2 One of the security features of a multi-user computer system is a user login process. The user must complete this successfully before they can access the resources of the system.

As part of the login process the user enters their user ID followed by a password. The system then compares the password entered with the password held in a file.

(a) The steps involved in the login process are described as follows:

- User enters their ID and password.
- Validation checks:
  - Compare user ID with data from the file.
  - Indicate whether or not the user ID was found.
  - If user ID found, check whether passwords match.

The description above is not detailed enough to allow a program to be written. The validation checks must be expressed as a more detailed algorithm.

Give the name of the process of increasing the level of detail of the algorithm.

.....[1]

(b) An identifier table is created as the algorithm is developed. A section of the table is shown. Complete the table.

Identifier	Data Type	Description
`UserIDInput`	.....	Stores the user ID entered
`PasswordInput`	.....	....
`UserIDFound`	.....	....
`PasswordValid`	.....	....

[5]

(c) The validation checks described in **part (a)** are to be used as the basis for program code.

Use **structured English** to write a more detailed algorithm.

You should:

- use the identifiers given in the previous table
- assume that the password file, `password.txt`, is organised as a simple text file. The user ID and password are stored together, one entry per line.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

3 A string conversion function, `StringClean`, is to be written.

This function will form a new string, `OutString`, from a given string, `InString`, by:

- removing all non-alphabetic characters
- converting all alphabetic characters to lower case.

For example:

```
InString = "Good Morning, Dave"
OutString = "goodmorningdave"
```

The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode using relevant built-in functions.

For the built-in functions list, refer to the **Appendix** on page 14.

```
FUNCTION StringClean(..........................) RETURNS ..........

   DECLARE NextChar : ....................................

   DECLARE ............................................... : STRING

   .................................... //initialise the return string

   //loop through InString to produce OutString

   FOR n ← 1 TO ............................. //from first to last

      NextChar ← ............................ //get next character and

      NextChar ← ............................ //convert to lower case

      IF ..................................... //check if alphabetic

         THEN

            ...................................... //add to OutString

      ENDIF

   ENDFOR

   ..........................................//return value

ENDFUNCTION
```

[11]

4 **(a)** A structure chart is a tool used in modular program design.

State **three** pieces of information that a structure chart can convey about a program design.

1 .....

.....

2 .....

.....

3 .....

.....
[3]

(b) The following diagram shows part of a structure chart.

Examples of the data items that correspond to the arrows are given in this table:

Arrow	Data item
A	`234.56`
B	`"Mr Robert Zimmerman"`
C	`True`

Use **pseudocode** to write the function header for the **Card payment** module.

.....

.....[3]

5 A multi-user computer system records user login information in a text file, `LoginFile.txt`. Each time a user successfully logs into the system, the following information is recorded:

Item	Information	Example data
1	A five character user ID	`"JimAA"`
2	A four character port ID	`"3456"`
3	A fourteen character time and date	`"08:30Jun012015"`

The data items are concatenated to form a single string. Each string is saved as a separate line in the text file.

The example data in the preceding table would result in the following text line in the file:

"JimAA345608:30Jun012015"

The computer system can produce a list of the successful login attempts by a given user.

The file `LoginFile.txt` is searched for a given user ID and the corresponding data are copied into a 2D array, `LoginEvents`.

`LoginEvents` has been declared in pseudocode as:

```
DECLARE LoginEvents[1 : 1000, 1 : 2] OF STRING
```

A procedure, `SearchFile`, is needed to search the file and copy selected data to the array.

The main steps of the procedure are as follows:

- Input a user ID.
- Search `LoginFile.txt` for entries with matching user ID.
- For matching entries, copy items 2 and 3 above into the `LoginEvents` array.

You can assume that:

- the system initialises all elements of `LoginEvents` to an empty string " ", before it calls `SearchFile`
- there will be no more than 1000 successful logins for a single user.

Write **program code** for the procedure `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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[10]

6 A multi-user computer system makes use of passwords.

To be valid, a password must comply with the following rules:

- at least two lower-case alphabetic characters
- at least two upper-case alphabetic characters
- at least three numeric characters
- alpha-numeric characters only

A function, `ValidatePassword`, is needed to check that a given password follows these rules. This function takes a string, `Pass`, as a parameter and returns a Boolean value:

- `TRUE` if `Pass` contains a valid password
- `FALSE` otherwise.

(a) Write **program code** to implement the new function `ValidatePassword`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[10]

(b) **(i)** The function will be tested.

Give a valid string to check that the function returns `TRUE` under the correct conditions.

String1: .....

Modify the valid string given for String1 to test each rule separately.

Explain your choice in each case.

String2: .....

Explanation: .....

.....

.....

String3: .....

Explanation: .....

.....

.....

String4: .....

Explanation: .....

.....

.....

String5: .....

Explanation: .....

.....

.....
[5]

(ii) When testing a module, it is necessary to test all possible paths through the code.

State the name given to this type of testing.

.....[1]

(iii) A program consisting of several modules may be tested using a process known as stub testing.

Explain this process.

.....

.....

.....

.....[2]

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

```
MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING
```

returns string of length `y` starting at position `x` from `ThisString`.
Example: **`MID("ABCDEFGH", 2, 3)`** returns string **`"BCD"`**

```
LENGTH(ThisString : STRING) RETURNS INTEGER
```

returns the integer value representing the length of string `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 string **`"ABC"`**

```
RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING
```

returns rightmost `x` characters from `ThisString`.
Example: **`RIGHT("ABCDEFGH", 3)`** returns string **`"FGH"`**

```
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 then it is returned unchanged.
Example: **`LCASE('W')`** returns **`'w'`**

```
MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```

returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`.
Example: **`MOD(10,3)`** returns **`1`**

```
DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```

returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`.
Example: **`DIV(10,3)`** returns **`3`**

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings. Example: **`"Summer" & " " & "Pudding"`** produces **`"Summer Pudding"`**
`AND`	Performs a logical **`AND`** of two Boolean values. Example: **`TRUE AND FALSE`** produces **`FALSE`**
`OR`	Performs a logical **`OR`** of two Boolean values. Example: **`TRUE OR FALSE`** produces **`TRUE`**

© UCLES 2017

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2017

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

* 5 5 0 9 2 4 3 6 5 1 *

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2015**

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 **19** printed pages and **1** blank page.

DC (LK/JG) 95392/5
© UCLES 2015

CAMBRIDGE International Examinations

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

1 A marathon runner records their time for a race in hours, minutes and seconds.

An algorithm is shown below in structured English.

INPUT race time as hours, minutes and seconds
CALCULATE race time in seconds
STORE race time in seconds
OUTPUT race time in seconds

(a) The identifier table needs to show the variables required to write a program for this algorithm.

Complete the table.

Identifier	Data type	Description
`RaceHours`	`INTEGER`	The hours part of the race time.

[3]

(b) Before the program is written, the design is amended.

The new design includes input of the runner's current personal best marathon time (in seconds).

The output will now also show one of the following messages:

- "Personal best time is unchanged"
- "New personal best time"
- "Equals personal best time"

(i) Show the additional variable needed for the new design.

Identifier	Data type	Description

[1]

(ii) Write **program code** for the new design.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[7]

(c) The program code will be tested using white-box testing.

(i) Explain what is meant by white-box testing.

.....

.....

.....[2]

(ii) Complete the table heading.
Complete Test Number 1.
Add the data for Test Number 2 and Test Number 3.

	Input values				**Output**	
Test number	**Race hours**	**Race minutes**	**Race seconds**	**.....**	**Total time (seconds)**	**Message**
1	3	4	13	11053	11053	
2				11053		
3				11053		

[6]

2 A program displays a menu with choices 1 to 4. The code to display the menu is written as the procedure `DisplayMenu`.

(a) Pseudocode which uses this procedure is:

```
CALL DisplayMenu
REPEAT
   OUTPUT "Enter choice (1..4)"
   INPUT Choice
UNTIL Choice >= 1 AND Choice <= 4
```

(i) Describe what this pseudocode will do.

.....

.....

.....

.....[3]

(ii) State why a loop is required.

.....

.....[1]

(b) The following pseudocode is a revised design.

```
CONSTANT i ← 3
CALL DisplayMenu
NoOfAttempts ← 0
REPEAT
   OUTPUT "Enter choice (1..4)"
   INPUT Choice
   NoOfAttempts ← NoOfAttempts + 1
UNTIL (Choice >= 1 AND Choice <= 4) OR NoOfAttempts = i
```

(i) Give the maximum number of inputs the user could be prompted to make.

..... [1]

(ii) State why this algorithm is an improvement on the one given in **part (a)**.

.....

.....[1]

(c) The pseudocode is in its initial stage of development.

The table below shows the action currently taken by the pseudocode following each menu choice.

Menu choice	Description	Program response
1	Read data from the customer file	Calls a procedure `ReadFile` which for testing purposes outputs the message “Read file code”
2	Add a customer	Outputs message “Add customer code”
3	Search for a customer	Outputs message “Search customer code”
4	Terminates the program	Ends

Complete the pseudocode for the design in **part (b)**, shown again below, to respond to each menu choice.

```
CONSTANT i ← 3
CALL DisplayMenu
NoOfAttempts ← 0
REPEAT
   OUTPUT "Enter choice (1..4)"
   INPUT Choice
   NoOfAttempts ← NoOfAttempts + 1
UNTIL (Choice >= 1 AND Choice <= 4) OR NoOfAttempts = i
```

.....

.....

.....

.....

.....

.....

.....[3]

(d) The algorithm in **part (c)** is to be amended. The program will:

- repeatedly display the menu and respond to the user's choice
- terminate when the user enters 4

Write **program code** for this final design which will be made up of:

- the main program
- procedure `ReadFile`
- procedure `DisplayMenu`

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

3 When the guarantee on a computer runs out, the owner can take out insurance to cover breakdown and repairs.

The price of the insurance is calculated from:

- the model of the computer
- the age of the computer
- the current insurance rates

Following an enquiry to the insurance company, the customer receives a quotation letter with the price of the insurance.

A program is to be produced.

The structure chart below shows the modular design for this process:

(a) Using the letters **A** to **D**, add the labelling to the chart boxes on the opposite page.

Modules	
A	Send quotation letter
B	Calculate price
C	Produce insurance quotation
D	Input computer details

[2]

(b) Using the letters **E** to **J**, complete the labelling on the chart opposite.

Some of these letters will be used more than once.

Data items	
E	`CustomerName`
F	`CustomerEmail`
G	`Model`
H	`Age`
I	`PolicyCharge`
J	`PolicyNumber`

[4]

4 A game is played between two players:

- they take turns at rolling a six-sided die (numbered 1 to 6) and record their throw
- a player scores 1 point if their throw is higher than their opponent
- they each roll the die 20 times
- if the player's throw is the same as their opponent, the total points is unchanged
- the winner is the player with the larger number of points after 20 throws

The pseudocode will use variable `NoOfThrows` as shown.

Identifier	Data type	Description
`NoOfThrows`	`INTEGER`	Loop control variable

(i) Complete the pseudocode for the given algorithm.

```
FOR ..............................................................................

  INPUT Player1Throw

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

  IF Player1Throw > Player2Throw
    THEN

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

  ENDIF
  IF Player2Throw > Player1Throw
    THEN
      Player2Total ← Player2Total + 1
  ENDIF

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

IF Player1Total > Player2Total
  THEN
    OUTPUT "Player1 is the winner"
  ELSE
    OUTPUT "Player2 is the winner"
ENDIF
```

[5]

(ii) Identify the game result which will produce incorrect output.

.....

.....[1]

Question 5 begins on page 12.

5 A company creates two new websites, Site X and Site Y, for selling bicycles.

Various programs are to be written to process the sales data.

These programs will use data about daily sales made from Site X (using variable `SalesX`) and Site Y (using variable `SalesY`).

Data for the first 28 days is shown below.

	`SalesDate`	`SalesX`	`SalesY`
1	03/06/2015	0	1
2	04/06/2015	1	2
3	05/06/2015	3	8
4	06/06/2015	0	0
5	07/06/2015	4	6
6	08/06/2015	4	4
7	09/06/2015	5	9
8	10/06/2015	11	9
9	11/06/2015	4	1
…			
28	01/07/2015	14	8

(a) Name the data structure to be used in a program for `SalesX`.

.....[2]

(b) The programmer writes a program from the following pseudocode design.

```
x ← 0
FOR DayNumber ← 1 TO 7
   IF SalesX[DayNumber] + SalesY[DayNumber] >= 10
      THEN
         x ← x + 1
         OUTPUT SalesDate[DayNumber]
   ENDIF
ENDFOR
OUTPUT x
```

(i) Trace the execution of this pseudocode by completing the trace table below.

x	DayNumber	OUTPUT
0		

[4]

(ii) Describe, in detail, what this algorithm does.

.....

.....

.....

.....[3]

(c) The company wants a program to output the total monthly sales for one of the selected websites.

The programmer codes a function with the following function header:

```
FUNCTION MonthlyWebSiteSales(ThisMonth : INTEGER, ThisSite : CHAR)
                                                    RETURNS INTEGER
```

The function returns the total number of bicycles sold for the given month and website.

The function will use the following:

Identifier	Data type	Description
`ThisMonth`	`INTEGER`	Represents the month number e.g. 4 represents April
`ThisSite`	`CHAR`	Coded as: • X for website X • Y for Website Y

(i) Give the number of parameters of this function. .....[1]

(ii) Some of the following function calls may be invalid.

Mark each call with:

- a tick (✓), for a valid call
- a cross (✗), for an invalid call

For any function calls which are invalid, explain why.

Function call	Tick (✓) / cross (✗)	Explanation (if invalid)
`MonthlyWebSiteSales(1, "Y")`		
`MonthlyWebSiteSales(11, 'X', 'Y')`		
`MonthlyWebSiteSales(12, 'X')`		

[3]

(d) The company decides to offer a discount on selected dates. A program is written to indicate the dates on which a discount is offered.

The program creates a text file, DISCOUNT_DATES (with data as shown), for a number of consecutive dates.

03/06/2015 TRUE
04/06/2015 FALSE
05/06/2015 FALSE
06/06/2015 FALSE
07/06/2015 FALSE
08/06/2015 FALSE
09/06/2015 FALSE
10/06/2015 TRUE
11/06/2015 FALSE
01/07/2015 FALSE

Each date and discount indicator is separated by a single <Space> character.

The discount indicators are:

- FALSE – indicates a date on which no discount is offered
- TRUE – indicates a date on which a discount is offered

A programming language has the built-in function `CONCAT` defined as follows:

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING] )
                                                    RETURNS STRING
```

For example:

```
CONCAT("San", "Francisco") returns "SanFrancisco"
CONCAT("New", "York", "City") returns "NewYorkCity"
```

The use of the square brackets indicates that the parameter is optional.

The following incomplete pseudocode creates the text file DISCOUNT_DATES.

Complete the pseudocode.

```
OPENFILE "DISCOUNT_DATES" FOR ..............................................................

INPUT ......................................................................................

WHILE  NextDate <>"XXX"

   INPUT Discount

   ..............................  = CONCAT(NextDate, " ", Discount)

   WRITEFILE "DISCOUNT_DATES",  NextLine

   INPUT NextDate

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

OUTPUT "File now created"

CLOSEFILE
```

[4]

Question 5(e) continues on page 18.

(e) The `DISCOUNT_DATES` text file is successfully created.

The company now wants a program to:

- key in a date entered by the user
- search the text file for this date
- if found, output one of the following messages:
  - "No discount on this date"
  - "This is a discount date"
- if not found, output "Date not found"

(i) Add to the identifier table to show the variables you need for this new program.

Identifier	Data type	Description
`DISCOUNT_DATES`	`FILE`	Text file to be used

[3]

(ii) Write the **program code**.

Do not include any declaration or comment statements for the variables used.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[7]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2015

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2023**

INSERT **2 hours**

3033251826-I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (DE) 328287
© UCLES 2023

An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

- A string of length 1 may be either of type `CHAR` or `STRING`
- A `CHAR` may be assigned to, or concatenated with, a `STRING`
- A `STRING` of length greater than 1 cannot be assigned to a `CHAR`

`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"`
`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 string `"BCD"`
`LENGTH(ThisString : STRING) RETURNS INTEGER` returns the integer value representing the length of `ThisString` Example: `LENGTH("Happy Days")` returns `10`

`TO_UPPER(x : <datatype>) RETURNS <datatype>`
`<datatype>` may be `CHAR` or `STRING`
returns an object of type `<datatype>` formed by converting all characters of `x` to upper case.
Examples:
• `TO_UPPER("Error 803")` returns `"ERROR 803"`
• `TO_UPPER('a')` returns `'A'`

`TO_LOWER(x : <datatype>) RETURNS <datatype>`
`<datatype>` may be `CHAR` or `STRING`
returns an object of type `<datatype>` formed by converting all characters of `x` to lower case.
Examples:
• `TO_LOWER("JIM 803")` returns `"jim 803"`
• `TO_LOWER('W')` returns `'w'`

`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>`
returns a string representation of a numeric value.
`<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING`
Example: `NUM_TO_STR(87.5)` returns `"87.5"`

`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>`
returns a numeric representation of a string.
`<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER`
Example: `STR_TO_NUM("23.45")` returns `23.45`

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN`
returns `TRUE` if `ThisString` represents a valid numeric value.
`<datatype>` may be `CHAR` or `STRING`
Example: `IS_NUM("-12.36")` returns `TRUE`

`ASC(ThisChar : CHAR) RETURNS INTEGER` returns an integer value (the ASCII value) of character `ThisChar` Example: `ASC('A')` returns `65`, `ASC('B')` returns `66`, etc.
`CHR(x : INTEGER) RETURNS CHAR` returns the character whose integer value (the ASCII value) is `x` Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'`, etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER` returns the integer part of `x` Example: `INT(27.5415)` returns `27`
`RAND(x : INTEGER) RETURNS REAL` returns a real number in the range `0` to `x` (**not** inclusive of `x`). Example: `RAND(87)` may return `35.43`

Date Functions

Date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER` returns the current day number from `ThisDate` Example: `DAY(04/10/2003)` returns `4`
`MONTH(ThisDate : DATE) RETURNS INTEGER` returns the current month number from `ThisDate` Example: `MONTH(04/10/2003)` returns `10`
`YEAR(ThisDate : DATE) RETURNS INTEGER` returns the current year number from `ThisDate` Example: `YEAR(04/10/2003)` returns `2003`
`DAYINDEX(ThisDate : DATE) RETURNS INTEGER` returns the day index number from `ThisDate` where Sunday = 1, Monday = 2 etc. Example: `DAYINDEX(09/05/2023)` returns 3
`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE` returns a value of type `DATE` with the value of `<Day>/<Month>/<Year>` Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26/10/2003
`TODAY() RETURNS DATE` returns a value of type `DATE` corresponding to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` The function will generate an error if the file is not already open in `READ` mode.

Operators

An error will be generated if an operator is used with a value or values of an incorrect type.

`&`	concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` The operator may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values. Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values. Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value. Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another. Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another. Example `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	used to compare two items of the same type.
	evaluates to `TRUE` if the condition is true, otherwise evaluates to `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when variable `Count` contains the value 4

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

8303835580

COMPUTER SCIENCE **9608/22**

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 **20** pages. Any blank pages are indicated.

DC (CJ) 205021/2
© UCLES 2021

BLANK PAGE

1 **(a)** Maintenance of programs may be needed for a number of different reasons.

State **two** types of maintenance **and** give a reason why each may be needed.

Type .....

Reason .....

.....

Type .....

Reason .....

.....
[4]

(b) State why characters need to be represented in ASCII or Unicode before they can be processed.

.....

..... [1]

(c) Each line of a text file contains several data items. A special character is inserted between data items before the line is written to the file.

Explain why a special character is used in this way.

.....

.....

.....

..... [2]

(d) 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 page 18 for the list of built-in pseudocode functions and operators.

Statement	Error
`Code ← RIGHT("Cap" & "art", 4)`	
`Status ← MID("Computer", 7, 5)`	
`Size ← LENGTH("Password") * 2`	
`NextChar ← CHR('A')`	
`Index ← Index & 3`	

[5]

2 Study the following pseudocode.

```
DECLARE Overload : BOOLEAN

PROCEDURE LEM()

  DECLARE Status : BOOLEAN
  DECLARE Landed : INTEGER

  Overload ← FALSE
  Landed ← FALSE

  WHILE Landed = FALSE
     Status ← Sample()
     IF Status = TRUE
        THEN
           Landed ← SubA(42)
        ELSE
           Overload ← SubB(37)
           IF Overload = TRUE
              THEN
                 CALL Display("Alarm 1202")
           ENDIF
     ENDIF
  ENDWHILE

ENDPROCEDURE
```

(a) Examine the pseudocode **and** complete the following table:

	Answer
The identifier name of a global variable	
The name of the loop structure	
The identifier involved in a data type mismatch	
The name of a procedure that takes a parameter	
The name of a function	

[5]

(b) Draw a program flowchart to represent the pseudocode algorithm.

Variable declarations are not required in program flowcharts.

[5]

3 **(a)** **(i)** Module names and parameters are features that may be represented on a structure chart.

State **two other** features than can be represented on a structure chart.

Feature 1 .....

Feature 2 .....
[2]

(ii) The headers for three modules in a program are defined in pseudocode as follows:

Pseudocode module header
`PROCEDURE Create(S2 : INTEGER, P3 : STRING)`
`PROCEDURE Modify(S2 : INTEGER, BYREF P4 : STRING)`
`FUNCTION Delete(P4 : INTEGER, M4 : STRING) RETURNS INTEGER`

A fourth module, `Membership()`, may call any one of the three modules.

Draw a structure chart to represent the information given about the **four** modules.

[5]

(b) Draw a diagram to show the stages of the program development cycle. Use arrows to indicate how the stages are linked.

[2]

4 **(a)** Using **pseudocode**, write a post-condition loop to output every odd number between 100 and 200.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) A program contains a global 2D array `XRef`. The array consists of 100 rows and 3 columns. The array is of data type `STRING`.

A function, `Search()`, takes two parameters `Par1` and `Par2` as string values and returns an integer value.

The function returns:

- the index number of the first row where:
  - the element in column 1 of that row matches `Par1` and
  - either of the other two elements in that row match `Par2`
- `-1` if no match found in any row.

Write **program code** for the function `Search()`.

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]

5 **(a)** An Integrated Development Environment (IDE) will be used to develop a program.

(i) An IDE includes features for program presentation.

State **two** of these presentation features.

Feature 1 .....

.....

Feature 2 .....

.....

[2]

(ii) Name **two** IDE features that can help with initial error detection.

Feature 1 .....

.....

Feature 2 .....

.....

[2]

(b) **(i)** A function, `Verify()`, is written in pseudocode.

Write the **two** missing lines to complete the pseudocode.

```
FUNCTION Verify(UserID : STRING) RETURNS BOOLEAN

  ..........................................
  DECLARE Password : STRING
  OUTPUT "Please Input your password: "
  INPUT Password
  Response ← Validate(UserID, Password) AND Today()

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

ENDFUNCTION
```

[2]

(ii) The function, `Verify()`, is to be amended as follows:

- if the `UserID` is `"Guest"`, a password is not required and `TRUE` should be returned
- output a message to try again if the password entered is not valid
- return `FALSE` if the number of attempts to enter a valid password exceeds three.

Write **program code** for the **amended** function `Verify()`.

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]

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)	`"JBCD0002"`	`""`
`Description`	`STRING`	a description of the item (alphabetic characters only)	`"soap"`	`""`
`Quantity`	`INTEGER`	the number in stock	`9`	`0`
`Cost`	`REAL`	the cost of the item	`1.45`	`0.0`

- Each array contains 10 000 elements.
- Elements with the same index relate to the same stock item. For example, `StockID[3]` contains the ID for the product whose description is in `Description[3]`.
- The `StockID` array is not sorted.

The program will be modified so that the data from the arrays can be stored in a text file for backup. You may assume that a backup file contains only valid stock data.

The programmer has started to define program modules as follows:

Module	Description

`Unpack()`	• called with two parameters:
	  ◦ an array index
	  ◦ a string value read from one line of the backup file
	• extracts the four data values from the string and assigns each to the appropriate array

`Restore()`	• called with a string representing the name of a backup file • returns `FALSE` if the file is empty • sets all elements of each array to the initial data value as given in the table • reads the backup file line by line calls `Unpack()` to extract data from each line and assign values to the corresponding arrays • returns `FALSE` if the arrays are full but there are still lines in the file, otherwise returns `TRUE`
`StockSummary()`	For all items where `StockID` does **not** contain the initial value: • counts the number of stock entries in the `StockID` array • outputs the overall value of all items in stock (cost multiplied by the quantity) • outputs the number of stock entries

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

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

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

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(c) The module description of `GetValidFilename()` is as follows:

Module	Description

`GetValidFilename()`	• prompts and inputs a filename
	• validates the filename by checking that it:
	  ◦ is between 4 and 10 characters in length (inclusive)
	  ◦ contains only alphanumeric characters
	• if the filename is invalid, outputs a warning message and asks the user to try again
	• otherwise returns the valid filename

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

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]

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`

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

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

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

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

4026563148

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2023**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (LK/SG) 315868/1
© UCLES 2023

Refer to the **insert** for the list of pseudocode functions and operators.

1 A shop sells car accessories. A customer order is created if an item cannot be supplied from current stock. A program is being developed to create and manage the customer orders.

(a) The following identifier table shows some of the data that will be stored for each order.

Complete the identifier table by adding meaningful variable names and appropriate data types.

Example value	Explanation	Variable name	Data type
`"Mr Khan"`	The name of the customer		
`3`	The number of items in the order		
`TRUE`	To indicate whether this is a new customer		
`15.75`	The deposit paid by the customer		

[4]

(b) Other variables in the program have example values as shown:

Variable	Example value
`Total`	`124.00`
`DepRate`	`2.00`
`Description`	`"AB12345:Cleaning Brush (small)"`

Complete the table by evaluating each expression using the example values.

Expression	Evaluates to
`(Total * DepRate) + 1.5`	
`RIGHT(Description, 7)`	
`(LENGTH(Description) - 8) > 16`	
`NUM_TO_STR(INT(DepRate * 10)) & '%'`	

[4]

(c) The data that needs to be stored for each customer order in part **(a)** is not all of the same type.

Describe an effective way of storing this data for many customer orders while the program is running.

.....

.....

.....

.....

.....

..... [3]

2 An algorithm will:

1. input a sequence of integer values, one at a time
2. ignore all values until the value 27 is input, then sum the remaining values in the sequence
3. stop summing values when the value 0 is input and then output the sum of the values.

(a) Draw a program flowchart to represent the algorithm.

[5]

(b) The solution to the algorithm includes iteration.

Give the name of a suitable loop structure that could be used.

Justify your answer.

Name .....

Justification .....

.....
[2]

3 The diagram represents a linked list Abstract Data Type (ADT).

- Ptr1 is the start pointer. Ptr2 is the free list pointer.
- Labels D40, D32, D11 and D100 represent the data items of nodes in the list.
- Labels F1, F2, F3 and F4 represent the data items of nodes in the free list.
- The symbol Ø represents a null pointer.

(a) The linked list is implemented using two variables and two 1D arrays as shown.

The pointer variables and the elements of the Pointer array store the indices (index numbers) of elements in the Data array.

Complete the diagram to show how the linked list as shown above may be represented using the variables and arrays.

Variable	Value
`Start_Pointer`	
`Free_List_Pointer`	5

Index	Data array	Pointer array
1	D32	2
2		3
3		
4	D40	
5		
6	F2	7
7		
8		

[5]

(b) The original linked list is to be modified. A new node D6 is inserted between nodes D32 and D11.

The algorithm required is expressed in four steps as shown.

Complete the steps.

1. Assign the data item ..... to ..... .
2. Set the ..... of this node to point to ..... .
3. Set Ptr2 to point to ..... .
4. Set pointer of ..... to point to ..... .

[4]

4 A procedure `Count()` will:

1. input a value (all values will be positive integers)
2. count the number of odd values and count the number of even values
3. repeat from step 1 until the value input is 99
4. output the two count values, with a suitable message.

The value 99 must not be counted.

(a) Write pseudocode for the procedure `Count()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) The procedure `Count()` is to be tested.

Typical test data would consist of odd and even values, for example:

23, 5, 64, 100, 2002, 1, 8, 900, 99

The purpose of this test would be to test a typical mix of even and odd values and check the totals.

Give **three** test data sequences that may be used to test **different** aspects of the procedure.

Do **not** include invalid data.

Sequence 1:

Test data .....

Purpose of test. .....

.....

Sequence 2:

Test data .....

Purpose of test. .....

.....

Sequence 3:

Test data .....

Purpose of test. .....

.....

[3]

5 A global 1D array of integers contains four elements, which are assigned values as shown:

```
Mix[1] ← 1
Mix[2] ← 3
Mix[3] ← 4
Mix[4] ← 2
```

A procedure `Process()` manipulates the values in the array.

The procedure is written in pseudocode:

```
PROCEDURE Process(Start : INTEGER)
   DECLARE Value, Index, Count : INTEGER

   Index ← Start
   Count ← 0

   REPEAT
      Value ← Mix[Index]
      Mix[Index] ← Mix[Index] - 1
      Index ← Value
      Count ← Count + 1
   UNTIL Count = 5

   Mix[4] ← Count * Index

ENDPROCEDURE
```

Complete the trace table on the opposite page by dry running the procedure when it is called as follows:

```
CALL Process(3)
```

Index	Value	Count	Mix[1]	Mix[2]	Mix[3]	Mix[4]

[6]

6 **(a)** A procedure `CreateFiles()` will take two parameters:

- a string representing a file name
- an integer representing the number of files to be created.

The procedure will create the number of text files specified.

Each file is given a different name. Each file name is formed by concatenating the file name with a suffix based on the file number. The suffix is always three characters.

For example, the call `CreateFiles("TestData", 3)` would result in the creation of the three files, TestData.001, TestData.002 and TestData.003.

Each file will contain a single line. For example, file TestData.002 would contain the string:

```
This is File TestData.002
```

Write pseudocode for `CreateFiles()`.

Assume both parameters are valid and that the integer value is between 1 and 999, inclusive.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) A module `CheckFiles()` will count the number of files produced by `CreateFiles()` in part **(a)**.

`CheckFiles()` will take a string representing a file name and return the number of files found.

(i) Identify the type of module that should be used for `CheckFiles()`.

..... [1]

(ii) Write the module header for `CheckFiles()`.

.....

..... [1]

(iii) State the file mode that should be used in `CheckFiles()`.

..... [1]

7 A program contains six modules:

Pseudocode module header
`PROCEDURE Module-A()`
`PROCEDURE Module-X(T1 : INTEGER, S2 : REAL)`
`PROCEDURE Reset(BYREF Code : INTEGER)`
`FUNCTION Restore(OldCode : INTEGER) RETURNS BOOLEAN`
`FUNCTION Module-Y(RA : INTEGER, RB : BOOLEAN) RETURNS BOOLEAN`
`FUNCTION Module-Z(SA : INTEGER) RETURNS INTEGER`

`Module-X()` calls `Reset()`
`Module-Y()` calls `Restore()`

(a) Complete the structure chart for these modules.

[4]

(b) Explain the meaning of the diamond symbol as used in the diagram in part **(a)**.

.....

..... [2]

BLANK PAGE

8 A class of students are developing a program to send data between computers. Many computers are connected together to form a wired network. Serial ports are used to connect one computer to another.

Each computer:

- is assigned a unique three-digit ID
- has three ports, each identified by an integer value
- is connected to between one and three other computers.

Data is sent as individual message strings.
Each string contains the destination ID (the ID of the computer that is to receive the message) followed by the data:

```
<DestinationID><Data>
```

Messages may pass through several computers on the way to their destination.
When a message arrives at a computer, that is **not** the destination, the program needs to forward it on to another computer using one of its serial ports.

The port to use is obtained from information that is stored in an array `RouteTable`.

`RouteTable` is a global 2D array of integers. It is declared in pseudocode as follows:

```
DECLARE RouteTable : ARRAY[1:6,1:3] OF INTEGER
```

The values in the first two columns of `RouteTable` define a range of ID values.
Column 3 gives the corresponding port number to use when forwarding the message to a computer with an ID within this range.

For example, the contents of `RouteTable` could be:

	Column 1	**Column 2**	**Column 3**
Row 1	100	199	1
Row 2	200	259	2
Row 3	−1	<undefined>	<undefined>
Row 4	260	399	2
Row 5	400	599	3
Row 6	600	999	1

In this example, a message that arrives with a `DestinationID` of "283" will be forwarded using port 2.

Row 3 in the example shows an unused row. These may occur anywhere. Unused rows have the column 1 element set to −1. The value of unused elements in the other two columns is undefined.

The programmer has defined the first program module as follows:

Module	Description
`GetPort()`	• takes a `DestinationID` as a parameter of type string • searches for the range corresponding to the `DestinationID` in the array • returns the port number, or returns −1 if no corresponding range is found

(a) Write pseudocode for module `GetPort()`.

Assume `DestinationID` contains a valid three-digit string.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) Copies of the same program will run on each computer. The program contains a global variable `MyID` of type string, which contains the unique ID of the computer in which the program is running.

When messages are received, they are placed on one of two stacks. Stack 1 is used for messages that have reached their destination and stack 2 is used for messages that will be forwarded on to another computer.

Additional modules are defined:

Module	Description

`StackMsg()`
(already written)
	• takes two parameters:
	  ○ a string representing a message
	  ○ an integer representing the stack number
	• adds the message to the required stack
	• returns `TRUE` if the message is added to the required stack, otherwise returns `FALSE`

`ProcessMsg()`	• takes a message as a parameter of type string • ignores any message with a zero-length data field • extract the `DestinationID` from the message • checks whether the `DestinationID` is this computer or whether the message is to be forwarded • uses `StackMsg()` to add the message to the appropriate stack • outputs an error if the message could not be added to the stack

Write pseudocode for module `ProcessMsg()`.

Module `StackMsg()` must be used.

..... [7]

(c) The program contains a module `GetFile()` which receives text files sent from another computer.

Lines from the file are sent one at a time. Each message contains one line and `ProcessMsg()` from part **(b)** adds each message as it is received onto stack 1.

Module `GetFile()` removes messages from stack 1 and writes the data to a text file.

There is a problem. Under certain circumstances, the received file does not appear as expected.

Assume that while a file is being received `ProcessMsg()` receives only messages containing lines from the file.

(i) Describe the circumstances and explain the problem.

Circumstances .....

.....

Explanation .....

.....

.....

.....
[3]

(ii) Suggest a more appropriate Abstract Data Type that could be used to store the messages that would not have the same problem.

..... [1]

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 Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2023

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

2390708169

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2021**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (PQ) 222284
© UCLES 2021

Refer to the **insert** for the list of pseudocode functions and operators.

1 Sylvia is testing a program that has been written by her colleague. Her colleague tells her that the program does not contain any syntax errors.

(a) **(i)** State what her colleague means by "does not contain any syntax errors".

.....

.....

..... [1]

(ii) Identify **and** describe **one** other type of error that the program may contain.

Type of error .....

Description .....

.....
[2]

(b) Complete the following table by giving the appropriate data type in each case.

Use of variable	Data type
The average mark in a class of 40 students	
An email address	
The number of students in the class	
To indicate whether an email has been read	

[4]

(c) An airline wants to provide passengers with information about individual flights and allow them to book their flight using an online booking system.

(i) Tick (✓) **one** box in each row of the table to indicate whether each item of information would be essential for the customer when making the booking.

Information	Essential	Not essential
Departure time		
Flight number		
Departure airport		
Aircraft type		
Ticket price		
Number of seats in aircraft		

[3]

(ii) Identify the technique used to filter out information that is not essential when designing the booking system **and** state one benefit of this technique.

Technique .....

Benefit .....

.....

[2]

(iii) Identify **two additional** pieces of essential information that a passenger might need when booking a flight.

1 .....

2 .....

[2]

BLANK PAGE

2 **(a)** An algorithm to sort a 1D array into ascending order is described as follows:

- move the largest value to the end
- keep repeating until the array is sorted.

Apply the process of stepwise refinement to this algorithm in order to produce a more detailed description.

Write the more detailed description using **structured English**. Your explanation of the algorithm should not include pseudocode statements.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) The program flowchart shown describes a simple algorithm.

Write pseudocode for the simple algorithm shown on page 6.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

3 **(a)** The diagram below represents a queue Abstract Data Type (ADT) that can hold a maximum of eight items.

The operation of this queue may be summarised as follows:

- The front of queue pointer points to the next item to be removed.
- The end of queue pointer points to the last item added.
- The queue is circular so that empty storage elements can be reused.

0	Frog	← Front of queue pointer
1	Cat	
2	Fish	
3	Elk	← End of queue pointer
4		
5		
6		
7		

(i) Describe how "Octopus" is added to the given queue.

.....

.....

.....

..... [2]

(ii) Describe how the next item in the given queue is removed and stored in the variable `AnimalName`.

.....

.....

.....

..... [2]

(iii) Describe the state of the queue when the **front of queue** and the **end of queue** pointers have the same value.

.....

..... [1]

(b) Some operations are carried out on the original queue given in **part (a)**.

(i) The current state of the queue is:

0	Frog
1	Cat
2	Fish
3	Elk
4	
5	
6	
7	

Complete the diagram to show the state of the queue after the following operations:

Add “Wasp”, “Bee” and “Mouse”, and then remove two data items.

[3]

(ii) The state of the queue after other operations are carried out is shown:

0	Frog	
1	Cat	
2	Fish	
3	Elk	← Front of queue pointer
4	Wasp	
5	Bee	
6	Mouse	← End of queue pointer
7	Ant	

Complete the following diagram to show the state of the queue after the following operations:

Remove one item, and then add “Dolphin” and “Shark”.

0	
1	
2	
3	
4	
5	
6	
7	

[2]

(c) The queue is implemented using a 1D array.

Describe the algorithm that should be used to modify the **end of queue pointer** when adding an item to the queue.

Your algorithm should detect any potential error conditions.

.....

.....

.....

.....

.....

..... [3]

4 A program controls the heating system in a sports hall.

Part of the program involves reading a value from a sensor. The sensor produces a numeric value that represents the temperature. The value is an integer, which should be in the range 0 to 40 inclusive.

A program function has been written to validate the values from the sensor.

(a) A test plan is needed to test the function.

Complete the table. The first line has been completed for you.

You can assume that the sensor will generate only integer data values.

Test	Test data value	Explanation	Expected outcome
1	23	Normal data	Data is accepted
2			
3			
4			
5			

[4]

(b) A program module controls the heaters. This module operates as follows:

- If the temperature is below 10, switch the heaters on.
- If the temperature is above 20, switch the heaters off.

Complete the following state-transition diagram for the heating system:

[3]

5 The following data items will be recorded each time a student successfully logs on to the school network:

Data item	Example data
Student ID	`"CJL404"`
Host ID	`"Lib01"`
Time and date	`"08:30, June 01, 2021"`

The Student ID is six characters long. The other two data items are of variable length.

A single string will be formed by concatenating the three data items. A separator character will need to be inserted between items two and three.

For example:

```
"CJL404Lib01<separator>08:30, June 01, 2021"
```

Each string represents one log entry.

A programmer decides to store the concatenated strings in a 1D array `LogArray` that contains 2000 elements. Unused array elements will contain an empty string.

(a) Suggest a suitable separator character **and** give a reason for your choice.

Character .....

Reason .....

.....
[2]

(b) The choice of data structure was made during one stage of the program development life cycle.

Identify this stage.

..... [1]

(c) A function `LogEvents()` will:

- take a Student ID as a parameter
- for each element in the array that matches the Student ID parameter:
  - add the value of the array element to the existing text file `LogFile`
  - assign an empty string to the array element
- count the number of lines added to the file
- return this count.

Write pseudocode for the function `LogEvents()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

6 A mobile phone has a touchscreen. The screen is represented by a grid, divided into 800 rows and 1280 columns.

The grid is represented by a 2D array `Screen` of type `INTEGER`. An array element will be set to 0 unless the user touches that part of the screen.

Many array elements are set to 1 by a single touch of a finger or a stylus.

The following diagram shows a simplified touchscreen. The dark line represents a touch to the screen. All grid elements that are wholly or partly inside the outline will be set to 1. These elements are shaded.
The element shaded in black represents the centre point.

A program is needed to find the coordinates (the row and column) of the centre point. The centre point on the diagram above is row 6, column 11.

Assume:

- the user may only touch one area at a time
- screen rotation does not affect the touchscreen.

The programmer has started to define program modules as follows:

Module	Description
`SetRow()` (generates test data)	• Called with three parameters of type `INTEGER`: ◦ a row number ◦ the number of pixels to be skipped starting from column 1 ◦ the number of pixels that should be set to 1 • Sets the required number of pixels to 1 For example, `SetRow(3, 8, 5)` will give row 3 as in the diagram shown.
`SearchInRow()`	• Takes two parameters of type `INTEGER`: ◦ a row number ◦ a start column (1 or 1280) • Searches the given row from the start column (either left to right or right to left) for the first column that contains an element set to 1 • Returns the column number of the first element in the given row that is set to 1 • Returns −1 if no element is set to 1
`SearchInCol()`	• Takes two parameters of type `INTEGER`: ◦ a column number ◦ a start row (1 or 800) • Searches the given column from the start row (either up or down) for the first row that contains an element set to 1 • Returns the row number of the first element in the given column that is set to 1 • Returns −1 if no element is set to 1

(a) Write pseudocode to implement the module `SetRow()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) The module description of `SearchInRow()` is provided here for reference.

Module	Description

`SearchInRow()`	• Takes two parameters of type `INTEGER`:
	  ◦ a row number
	  ◦ a start column (1 or 1280)
	• Searches the given row from the start column (either left to right or right to left) for the first column that contains an element set to 1
	• Returns the column number of the first element in the given row that is set to 1
	• Returns −1 if no element is set to 1

Write pseudocode to implement the module `SearchInRow()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(c) The following new module is introduced:

Module	Description
`GetCentreCol()`	• Called with a row number as an `INTEGER` • Uses `SearchInRow()` to find the first and last columns in the given row which have an array element set to 1 • Returns the index of the column midway between the first and last columns • Returns −1 if no element is set to 1

Write pseudocode to implement the module `GetCentreCol()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

BLANK PAGE

BLANK PAGE

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2021**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (CE/CGW) 213696/3
© UCLES 2021

Refer to the **insert** for the list of pseudocode functions and operators.

1 Sylvia is testing a program that has been written by her colleague. Her colleague tells her that the program does not contain any syntax errors.

(a) **(i)** State what her colleague means by "does not contain any syntax errors".

.....

.....

..... [1]

(ii) Identify **and** describe **one** other type of error that the program may contain.

Type of error .....

Description .....

.....
[2]

(b) Complete the following table by giving the appropriate data type in each case.

Use of variable	Data type
The average mark in a class of 40 students	
An email address	
The number of students in the class	
To indicate whether an email has been read	

[4]

(c) An airline wants to provide passengers with information about individual flights and allow them to book their flight using an online booking system.

(i) Tick (✓) **one** box in each row of the table to indicate whether each item of information would be essential for the customer when making the booking.

Information	Essential	Not essential
Departure time		
Flight number		
Departure airport		
Aircraft type		
Ticket price		
Number of seats in aircraft		

[3]

(ii) Identify the technique used to filter out information that is not essential when designing the booking system **and** state one benefit of this technique.

Technique .....

Benefit .....

.....

[2]

(iii) Identify **two additional** pieces of essential information that a passenger might need when booking a flight.

1 .....

2 .....

[2]

BLANK PAGE

2 **(a)** An algorithm to sort a 1D array into ascending order is described as follows:

- move the largest value to the end
- keep repeating until the array is sorted.

Apply the process of stepwise refinement to this algorithm in order to produce a more detailed description.

Write the more detailed description using **structured English**. Your explanation of the algorithm should not include pseudocode statements.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) The program flowchart shown describes a simple algorithm.

Write pseudocode for the simple algorithm shown on page 6.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

3 **(a)** The diagram below represents a queue Abstract Data Type (ADT) that can hold a maximum of eight items.

The operation of this queue may be summarised as follows:

- The front of queue pointer points to the next item to be removed.
- The end of queue pointer points to the last item added.
- The queue is circular so that empty storage elements can be reused.

0	Frog	← Front of queue pointer
1	Cat	
2	Fish	
3	Elk	← End of queue pointer
4		
5		
6		
7		

(i) Describe how "Octopus" is added to the given queue.

.....

.....

.....

..... [2]

(ii) Describe how the next item in the given queue is removed and stored in the variable `AnimalName`.

.....

.....

.....

..... [2]

(iii) Describe the state of the queue when the **front of queue** and the **end of queue** pointers have the same value.

.....

..... [1]

(b) Some operations are carried out on the original queue given in **part (a)**.

(i) The current state of the queue is:

0	Frog
1	Cat
2	Fish
3	Elk
4	
5	
6	
7	

Complete the diagram to show the state of the queue after the following operations:

Add “Wasp”, “Bee” and “Mouse”, and then remove two data items.

[3]

(ii) The state of the queue after other operations are carried out is shown:

0	Frog	
1	Cat	
2	Fish	
3	Elk	← Front of queue pointer
4	Wasp	
5	Bee	
6	Mouse	← End of queue pointer
7	Ant	

Complete the following diagram to show the state of the queue after the following operations:

Remove one item, and then add “Dolphin” and “Shark”.

0	
1	
2	
3	
4	
5	
6	
7	

[2]

(c) The queue is implemented using a 1D array.

Describe the algorithm that should be used to modify the **end of queue pointer** when adding an item to the queue.

Your algorithm should detect any potential error conditions.

.....

.....

.....

.....

.....

..... [3]

4 A program controls the heating system in a sports hall.

Part of the program involves reading a value from a sensor. The sensor produces a numeric value that represents the temperature. The value is an integer, which should be in the range 0 to 40 inclusive.

A program function has been written to validate the values from the sensor.

(a) A test plan is needed to test the function.

Complete the table. The first line has been completed for you.

You can assume that the sensor will generate only integer data values.

Test	Test data value	Explanation	Expected outcome
1	23	Normal data	Data is accepted
2			
3			
4			
5			

[4]

(b) A program module controls the heaters. This module operates as follows:

- If the temperature is below 10, switch the heaters on.
- If the temperature is above 20, switch the heaters off.

Complete the following state-transition diagram for the heating system:

[3]

5 The following data items will be recorded each time a student successfully logs on to the school network:

Data item	Example data
Student ID	`"CJL404"`
Host ID	`"Lib01"`
Time and date	`"08:30, June 01, 2021"`

The Student ID is six characters long. The other two data items are of variable length.

A single string will be formed by concatenating the three data items. A separator character will need to be inserted between items two and three.

For example:

```
"CJL404Lib01<separator>08:30, June 01, 2021"
```

Each string represents one log entry.

A programmer decides to store the concatenated strings in a 1D array `LogArray` that contains 2000 elements. Unused array elements will contain an empty string.

(a) Suggest a suitable separator character **and** give a reason for your choice.

Character .....

Reason .....

.....
[2]

(b) The choice of data structure was made during one stage of the program development life cycle.

Identify this stage.

..... [1]

(c) A function `LogEvents()` will:

- take a Student ID as a parameter
- for each element in the array that matches the Student ID parameter:
  - add the value of the array element to the existing text file `LogFile`
  - assign an empty string to the array element
- count the number of lines added to the file
- return this count.

Write pseudocode for the function `LogEvents()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

6 A mobile phone has a touchscreen. The screen is represented by a grid, divided into 800 rows and 1280 columns.

The grid is represented by a 2D array `Screen` of type `INTEGER`. An array element will be set to 0 unless the user touches that part of the screen.

Many array elements are set to 1 by a single touch of a finger or a stylus.

The following diagram shows a simplified touchscreen. The dark line represents a touch to the screen. All grid elements that are wholly or partly inside the outline will be set to 1. These elements are shaded.
The element shaded in black represents the centre point.

A program is needed to find the coordinates (the row and column) of the centre point. The centre point on the diagram above is row 6, column 11.

Assume:

- the user may only touch one area at a time
- screen rotation does not affect the touchscreen.

The programmer has started to define program modules as follows:

Module	Description
`SetRow()` (generates test data)	• Called with three parameters of type `INTEGER`: ◦ a row number ◦ the number of pixels to be skipped starting from column 1 ◦ the number of pixels that should be set to 1 • Sets the required number of pixels to 1 For example, `SetRow(3, 8, 5)` will give row 3 as in the diagram shown.
`SearchInRow()`	• Takes two parameters of type `INTEGER`: ◦ a row number ◦ a start column (1 or 1280) • Searches the given row from the start column (either left to right or right to left) for the first column that contains an element set to 1 • Returns the column number of the first element in the given row that is set to 1 • Returns −1 if no element is set to 1
`SearchInCol()`	• Takes two parameters of type `INTEGER`: ◦ a column number ◦ a start row (1 or 800) • Searches the given column from the start row (either up or down) for the first row that contains an element set to 1 • Returns the row number of the first element in the given column that is set to 1 • Returns −1 if no element is set to 1

(a) Write pseudocode to implement the module `SetRow()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) The module description of `SearchInRow()` is provided here for reference.

Module	Description

`SearchInRow()`	• Takes two parameters of type `INTEGER`:
	  ◦ a row number
	  ◦ a start column (1 or 1280)
	• Searches the given row from the start column (either left to right or right to left) for the first column that contains an element set to 1
	• Returns the column number of the first element in the given row that is set to 1
	• Returns −1 if no element is set to 1

Write pseudocode to implement the module `SearchInRow()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(c) The following new module is introduced:

Module	Description
`GetCentreCol()`	• Called with a row number as an `INTEGER` • Uses `SearchInRow()` to find the first and last columns in the given row which have an array element set to 1 • Returns the index of the column midway between the first and last columns • Returns −1 if no element is set to 1

Write pseudocode to implement the module `GetCentreCol()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

BLANK PAGE

BLANK PAGE

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.

© UCLES 2021

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2021**

INSERT **2 hours**

9601387009–I

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

DC (LK) 213666/2
© UCLES 2021

Note: An error occurs if a function call is not properly formed, or if the parameters are incorrect.

STRING Functions

`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"`

`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`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the lower case equivalent of `ThisChar`
Alphabetic characters that are not upper case are unchanged.

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

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the upper case equivalent of `ThisChar`
Alphabetic characters that are not lower case are unchanged.

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

`TO_UPPER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to upper case.

Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`

`TO_LOWER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to lower case.

Example: `TO_LOWER("JIM 803")` returns `"jim 803"`

© UCLES 2021

```
NUM_TO_STR(x : <data type1>) RETURNS <data type2>
```
returns a string representation of a numeric value.

Note: `<data type1>` may be `REAL` or `INTEGER`
`<data type2>` may be `CHAR` or `STRING`

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

```
STR_TO_NUM(x : <data type1>) RETURNS <data type2>
```
returns a numeric representation of a string.

Note: `<data type1>` may be `CHAR` or `STRING`
`<data type2>` may be `REAL` or `INTEGER`

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

```
IS_NUM(ThisString : STRING) RETURNS BOOLEAN
```
returns the value `TRUE` if `ThisString` represents a valid numeric value.

Example 1: `IS_NUM("12.36")` returns `TRUE`
Example 2: `IS_NUM("-12.36")` returns `TRUE`
Example 3: `IS_NUM("12.3a")` returns `FALSE`

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

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

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

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

NUMERIC Functions

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

Example: `INT(27.5415)` returns `27`

```
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`

OTHER Functions

`EOF(FileName : STRING) RETURNS BOOLEAN` returns `TRUE` if there are no more lines to be read from file `FileName` Note: The function will generate an error if the file is not already open in `READ` mode.

Note: An error occurs if an operator with a value of an incorrect type is used.

OPERATORS

`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to "Summer Pudding"
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	Performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	Finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	Finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

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.

© UCLES 2021

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER				

6264905153

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2018**

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 **15** printed pages and **1** blank page.

DC (LK/CGW) 146140/3
© UCLES 2018

CAMBRIDGE International Examinations

Question 1 begins on the next page.

© UCLES 2018

1 **(a)** A program stores data about hospital patients.

Give a suitable **identifier name** for each of the data items.

Description of data item	Suitable identifier name
The temperature of the patient	
The temperature of the room	
The patient identification number	
The name of the nurse taking the measurement	

[4]

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

Variable	Value
`MyGreeting`	`"Happy Birthday"`
`MyInitial`	`'C'`
`AgeInYears`	`27`
`Weight`	`60.5`
`Married`	`TRUE`
`Children`	`TRUE`

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 15.

Expression	Evaluates to
`"Mon" & MID(MyGreeting, 10, 2)`	
`AgeInYears + ASC(MyInitial)`	
`INT(MyInitial)`	
`MOD(Weight * 2, 10)`	
`Married AND (NOT Children)`	

[5]

(ii) Programming languages support different data types.

Give an appropriate data type for each of these variables from **part (b)(i)**.

Variable	Data type
`MyGreeting`	
`MyInitial`	
`AgeInYears`	
`Weight`	
`Married`	

[5]

2 The following is a function design in pseudocode.

Line numbers are given for reference only.

```
01  FUNCTION StringClean(InString : STRING) RETURNS STRING
02
03     DECLARE NextChar : CHAR
04     DECLARE OutString : STRING
05     DECLARE Counter : INTEGER
06     DECLARE MyString : STRING
07
08     OutString ← ""
09
10     FOR Counter ← 1 TO LENGTH(InString)
11
12        NextChar ← MID(InString,Counter,1)
13        NextChar ← LCASE(NextChar)
14
15        IF (NextChar >= 'a') AND (NextChar <= 'z')
16
17           THEN
18
19              OutString ← OutString & NextChar
20
21        ENDIF
22
23     ENDFOR
24
25     RETURN OutString
26
27  ENDFUNCTION
```

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

State **four** such features.

Feature 1 .....

Feature 2 .....

Feature 3 .....

Feature 4 .....

[4]

(ii) State **one** feature that could be added to make the pseudocode easier to understand.

.....[1]

(b) Study the function `StringClean()`. Identify the features of the function in the following table.

Feature	Answer
A line number containing an example of an assignment statement	
A line number containing the start of a repetition block	
A line number containing the end of a repetition block	
A line number containing the start of a selection statement	
The number of parameters of the MID function	
The Boolean operator used	
The number of local variables	
The number of function calls from within `StringClean()` resulting from the call: `NewString ← StringClean("Me")`	
The number of a line containing an unnecessary statement	

[9]

3 In a chemical factory, a procedure, `CheckSensor()` is required to allow an operator to monitor the temperature in different locations.

In the factory:

- the temperature is measured by 10 sensors, each at a different location
- each sensor has a unique ID (1 to 10).

The procedure `CheckSensor()` will compare the measured temperature against each of two constant values, `LowTemp` and `HighTemp`. It will perform the following actions depending on the result of the comparison.

Measured temperature	Action
below `LowTemp`	Output `"Cold"`
from `LowTemp` to `HighTemp`	Output `"Normal"`
above `HighTemp`	Call procedure `Alarm()`

A library function, `GetTemp()`, returns the temperature value from a given sensor.

The structured English representing the algorithm for the procedure `CheckSensor()` is as follows:

1. Prompt for the input of a sensor ID.
2. Input a sensor ID.
3. If the sensor ID is invalid, repeat from step 1.
4. Call the `GetTemp()` function with the sensor ID as the parameter, to obtain the relevant temperature.
5. Compare the temperature against the two constant values and take the appropriate action.

Draw a program flowchart on the next page to represent the algorithm for procedure `CheckSensor()`.

Variable declarations are not required in program flowcharts.

[8]

4 **(a)** A structure chart is used in modular program design.

Iteration and selection are two features of an algorithm that may be shown on a structure chart.

Give **three** other features.

Feature 1 .....

.....

Feature 2 .....

.....

Feature 3 .....

.....

[3]

(b) Pseudocode for a function is shown.

```
FUNCTION ItemProcess (AddItem, InString : STRING) RETURNS BOOLEAN
  DECLARE RetFlag : BOOLEAN
  RetFlag ← FALSE

  IF AddItem = "Yes"
     THEN
        RetFlag ← AddToList(InString)
     ELSE
        CALL RemoveFromList(InString)
  ENDIF

  RETURN RetFlag

ENDFUNCTION
```

Draw a structure chart on the next page to represent this pseudocode.

[6]

5 A golf club holds information about its members. When a member completes a round of golf, their score is stored along with their membership number and the date of the round.

(a) Explain why the club stores these data in a file rather than an array.

.....

.....[1]

(b) Editing functions such as cut, copy and paste are features provided by an Integrated Development Environment (IDE).

Give **two** additional features of an IDE that are helpful when **coding** a program.

Feature 1 .....

.....

Feature 2 .....

.....
[2]

(c) The information is stored in a text file, `ScoreDetails.txt`. The format of each line of the text file is as follows:

```
<MembershipNumber><Date><Score>
```

- `MembershipNumber` is a four-digit numeric string.
- `Date` is a six-digit numeric string in the format DDMMYY
- `Score` is a two-digit numeric string in the range "`50`" to "`99`".

A procedure, `AddNewScores()`, is being developed. This will allow the user to enter scores for several members on a particular date.

The procedure, `AddNewScores()`, will perform the following actions:

1. Prompt for the date of the scores.
2. Input the date of the scores.
3. Prompt for the membership number.
4. Input the membership number.
5. If the membership number is an empty string then end the procedure.
6. Prompt for the score.
7. Input the score.
8. Validate the score.
9. If the validation fails then repeat from step 6.
10. Form a text string from the data and write this to the `ScoreDetails.txt` file.
11. Repeat from step 3.

Write **program code** for the `AddNewScores()` procedure.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[9]

6 **(a)** The following pseudocode includes references to a 1D array.

```
DECLARE StudentGrade : ARRAY[1:5] OF CHAR
DECLARE n : INTEGER
DECLARE x : CHAR

n ← 3
x ← StudentGrade[n]
```

(i) Use the correct technical terms to explain the meaning of `[1:5]` in this pseudocode.

.....

.....

.....

.....[2]

(ii) Use the correct technical term to complete the following statement.

Integer `n` is used as the ..... to `StudentGrade`. [1]

(b) A 2D array, `Picture`, contains data representing a bitmap image. Each element of the array represents one pixel of the image. The image is grey-scale encoded where the value of each pixel ranges from 0 (representing black) to 255 (representing white) with intermediate values representing different levels of grey.

The following is an example of an image and the corresponding data values for the `Picture` array.

Bitmap image

Values

240	10	10	10	10	10	10	240
80	80	240	80	80	240	80	80
80	80	240	80	80	240	80	80
80	80	150	150	150	150	80	80
80	80	240	240	240	240	80	80
80	80	150	150	150	150	80	80
240	240	150	150	150	150	240	240
240	240	150	150	150	150	240	240

In pseudocode, the array is declared as follows:

```
DECLARE Picture : ARRAY[1:8, 1:8] OF INTEGER
```

A function, `Lighten()`, is required to lighten the image. Lightening an image may cause it to 'burn out'. An image is said to be 'burnt out' if any pixel is set to the maximum value of 255.

The function `Lighten()` will:

1. increase the value of each pixel by 10%
2. return TRUE if the resultant image is 'burnt out'.

Write **pseudocode** to implement the `Lighten()` function.

Assume that the array `Picture` is a global variable.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

7 A function, `ProcessMarks()`, is required to analyse test marks for a class of students.

- There are 20 students in the class.
- A mark is between 0 and 100.
- The marks for the class are stored in an array, `Mark`, which has 20 elements.
- The array is passed to the function as a parameter.
- The function will output a message stating the average and highest marks. For example:
  `"The average mark is 34 and the highest mark is 76"`
- The function returns the subscript of the highest mark.

Write **program code** to implement the `ProcessMarks()` function.

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 string `"BCD"`

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

`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 then it is returned unchanged.
Example: `LCASE('W')` returns `'w'`

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

Example: `INT(27.5415)` returns `27`

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

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

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

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`

© UCLES 2018

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2018

Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

2463170706

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2016**

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 **19** printed pages and **1** blank page.

DC (NF/SG) 115874/3
© UCLES 2016

CAMBRIDGE International Examinations

There is an **Appendix** on pages 18 and 19. Some questions will refer you to this information.

1 A number of players take part in a competition. The competition consists of a number of games. Each game is between two players. The outcome of a game is that each player is awarded a grade (A, B, C or D). Each grade has an associated number of points as shown in the table below.

Grade	Points
A	0
B	1
C	3
D	5

The points total for all players is recorded. After each game is completed, the total number of points for each player is updated.

For example:

- before the game between Ryan and Karina, Ryan's total is 5 points and Karina's total is 3 points
- the result of the game between Ryan and Karina is: Ryan achieved grade B, Karina achieved grade D
- the players' points totals are updated to: Ryan has 6 and Karina has 8

When a player's points total reaches 12 or higher, that player is removed from the competition.

A programmer will write a program to update the player total after a game.

The program will output:

- the player's updated points total
- the message 'ELIMINATED' if the player is removed from the competition.

The programmer designs the identifier table below:

Identifier	Data type	Description
`PlayerName`	`STRING`	Name of the player
`PlayerGameGrade`	`CHAR`	Game grade for the player
`PointsTotal`	`INTEGER`	Current player points
`SavePlayerTotal`	procedure	Procedure has parameters `PlayerName` and `PointsTotal` and saves the updated player total
`ReadPlayerTotal`	function	Function has a parameter `PlayerName` and returns the current total for that player

(a) Complete the following program flowchart by:

- filling in the boxes, using pseudocode where appropriate
- labelling the lines of the flowchart, where necessary.

[9]

Start
Yes
No
Stop

(b) Test data is to be produced to test the flowchart.

Complete the table of test data below to show **five** tests that should be used to test different paths through the flowchart.

Test data		**Expected results**	
`PointsTotal`	`PlayerGameGrade`	Updated `PointsTotal`	Output

[5]

(c) The programmer amends the design to validate the value of player game grade that the user inputs.

The amended part of the flowchart is shown below.

Write the equivalent **pseudocode** using a pre-condition loop, for this part of the amended flowchart.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

2 You will need to refer to the list of pseudocode string-handling functions in the **Appendix**.

(a) Give the value of the variables `x`, `y` and `z` for the following sequence of statements.

(i)
```
x ← ONECHAR("Barcelona", 5)
```
x ..... [1]

(ii)
```
y ← ONECHAR("Cool", 1) & ONECHAR("Ball", 2) & "t-food"
```
y ..... [1]

(iii)
```
Temp1 ← "13"
Temp2 ← ONECHAR("One-2-One", 5)
z ← TONUM(Temp2 & Temp1)
```
z ..... [1]

A computer program is to simulate the reading and processing of a string of characters from an input device.

The character string consists of:

- a number of digit characters
- one or more <*> characters, each used as a separator
- a final <#> character.

A typical input character sequence, stored as `InputString` is:

```
13*156*9*86*1463*18*#
```

Study this pseudocode.

```
01 DECLARE Numbers ARRAY [1:100] OF INTEGER
02 DECLARE InputString      : STRING
03 DECLARE NextChar         : CHAR
04 DECLARE NextNumberString : STRING
05 DECLARE i                : INTEGER    // Numbers array index
06 DECLARE j                : INTEGER    // InputString index
07
08 OUTPUT "String ... "
09 INPUT InputString
10 j ← 1
11 NextChar ← ONECHAR(InputString, j)
12
13 i ← 1
14 WHILE NextChar <> '#'
15   NextNumberString = ""
16   WHILE NextChar <> '*'
17     NextNumberString ← NextNumberString & NextChar
18     j ← j + 1
19     NextChar ← ONECHAR(InputString, j)
20   ENDWHILE
21
22   // store the next integer to the array
23   Numbers[i] ← TONUM(NextNumberString)
24   i ← i + 1
25   j ← j + 1
26   NextChar ← ONECHAR(InputString, j)
27 ENDWHILE
28
29 CALL DisplayArray()
```

(b) Write the line number for:

(i) A statement which declares a global variable used to store a single character. ..... [1]

(ii) A statement which runs code written as a procedure. ..... [1]

(iii) A statement which indicates the start of a 'pre-condition' loop. ..... [1]

(iv) A statement which increments a variable. ..... [1]

(c) Copy the condition which is used to control the inner loop.

..... [1]

(d) **(i)** Complete the trace table below for the given pseudocode as far as line 27.

The input string is: `23*731*5*#`

i	j	NextChar	NextNumberString	Numbers		
				1	2	3
1	1	'2'				
			""			
			"2"			
	2	'3'	"23"			
	3	'*'		23		

[5]

(ii) Explain what this algorithm does.

.....

.....

.....

..... [2]

3 Radhika mostly studied the high-level programming language XYZ at university.
She has been working in her first job for two years using language XYZ.
She applied for a new job which stated:

"The majority of the development work is done using language ABC."

(a) Radhika was interviewed for the job. Part of the interview process was to study some program code written in language ABC.

```
11 settype($TimesTable, Integer);
12 settype($upTo, Integer);
13 settype($Posn, Integer);
14 settype($Product, Integer);
15 $TimesTable = 7;
16 $UpTo = 10;
17
18 $Posn = 1
19 While ($Posn < $UpTo + 1)
20 {
21 $Product = $Posn * $TimesTable;
22 Echo $Posn . ' X' . $TimesTable . ' = ' . $Product . "<br>";
23 $Posn = $Posn + 1;
24 }
```

Answer the following questions taken from the interview.

(i) State what the `settype` keyword does in this language.

.....

..... [1]

(ii) Name **one** variable that the code uses.

..... [1]

(iii) Give a line number for an assignment statement.

..... [1]

(iv) Line `19` is the start of a pre-condition loop.

State the syntax that language ABC uses to indicate which statements must be executed inside a loop.

..... [1]

(b) **(i)** Describe what is meant by a transferable skill.

.....

.....

..... [2]

(ii) Give **two** examples which suggest that programming in a high-level language is a transferrable skill.

1 .....

.....

2 .....

..... [2]

4 A social club runs a weekly prize draw. A member can buy a ticket for $2. Each week, the club sells up to 150 tickets, with consecutive numbers starting from 1. There is a main cash prize of $120 and a further three prizes of $10. The computer program will generate the winning numbers.

Throughout this question, you will need to refer to the list of pseudocode functions in the **Appendix**.

(a) Write **pseudocode** to show how the `RND()` function can be used to generate a single integer in the range 1 to 150.

.....

..... [3]

(b) Write **program code** to generate and output four winning numbers.
Ignore the issue that duplicate numbers may be generated.

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

.....

.....

.....

.....

.....

.....

..... [4]

(c) The prize draw is a success. After six months, the club decided to sell an unlimited number of tickets each week. In any week, the total number of prizes may vary.

The programmer modifies the code written in **part(b)**.

The revised program will use a function to generate a single winning number. Therefore, in a week when the club offers six prizes, the program will use the function six times to generate the six winning numbers.

The function has identifier `GenerateNumber` and:

- has a single parameter for the number of tickets sold that week
- returns a single winning number.

Write **program code** for the function header.

Programming language .....

.....

..... [3]

(d) The organisers do not want the same number to be drawn more than once in a week. To deal with the issue of duplicate numbers being generated, the program in **part (b)** will require adaptive maintenance.

(i) Describe what is meant by adaptive maintenance.

.....

.....

.....

..... [2]

(ii) The club does not want the program to output the same number more than once in a week.

For each winning number, outline the extra steps that the program must have. Include any data structure that the program needs.
Do not write pseudocode or program code.

.....

.....

.....

.....

..... [3]

5 A team keeps a record of the scores made by each of their eight players in a number of games.

The data in the two tables below shows:

- the scores of the eight players after twenty games
- the eight player names.

	1	2	3		8
1	12	17	67		31
2	35	82	44		29
3	61	39	80		17
4	81	103	21		11
5	56	0	98		4
…					
19	45	6	81		77
20	12	11	3		6

1	Vorma
2	Ravi
3	Chada
4	Nigam
5	Bahri
6	Smith
7	Goyal
8	Lata

The team wants a computer program to input and record the player data.

(a) A programmer designs the following pseudocode for the input of a player's score from one game.

```
INPUT GameNumber
INPUT PlayerNumber
INPUT PlayerGameScore
PlayerScore[GameNumber, PlayerNumber] ← PlayerGameScore
```

Describe the data structure the programmer has used for the storage of all player scores.

..... [2]

(b) The player names are permanently stored in a text file `NAMES.TXT`, with one name on each line. The player names will be read by the program and stored in a 1D array.

The design given in **part (a)** will be expanded so that the user is prompted for the player name instead of the player number. Step `02` now becomes:

`02.1` Read the player names from file `NAMES.TXT` into the array `PlayerName`
`02.2` `INPUT ThisPlayerName`
`02.3` Search the `PlayerName` array for `ThisPlayerName` to find the `PlayerNumber`

(i) State the computing term for the expansion of one or more steps in the original design.

..... [1]

(ii) Write the **program code** for step `02.1`

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(iii) Program code is to be designed and written for step `02.3`

The program will use these identifiers:

Identifier	Data type	Description
`PlayerName`	`ARRAY[1 : 8] OF STRING`	Stores the player names (read from the file)
`ThisPlayerName`	`STRING`	Input by the user (step `02.2`)
`Found`	`BOOLEAN`	Flags when `ThisPlayerName` is found when searching the `PlayerName` array
`i`	`INTEGER`	Array index

Write **program code** to carry out the linear search for step `02.3`

There is no requirement to declare or comment on variables used.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) The team wants the program to produce a report, with the following specification.

The program outputs the total number of player scores that are:

- 50 and over but less than 100
- 100 or higher.

You can assume that before the section runs, the program has assigned all eight player scores to the `PlayerScore` data structure.

A first attempt at the pseudocode is shown below:

```
01 Total50 ← 0
02 Total100 ← 0
03 FOR PlayerIndex  ← 1 TO 8
04   FOR GameIndex ← 1 TO 20
05     IF PlayerScore[GameIndex, PlayerIndex] > 100
06       THEN
07         Total100 ← Total100 + 1
08       ELSE
09         IF PlayerScore[GameIndex, PlayerIndex] > 50
10           THEN
11             Total50 ← Total50 + GameIndex
12         ENDIF
13     ENDIF
14   ENDFOR
15 ENDFOR
16 OUTPUT Total50
17 OUTPUT Total100
```

(i) Describe the control structure used in lines `03` and `04` and lines `14` and `15`.

.....

.....

..... [2]

(ii) Consider the following two statements.

Write either TRUE **or** FALSE next to each statement.

Statement	TRUE or FALSE
The pseudocode considers all the scores for a player, before progressing to the next player.	
The pseudocode considers all scores in a game, before progressing to the next game.	

[1]

(iii) The programmer has made logic errors in the design.

State a line number at which an error occurs.

Explain the error or write the corrected pseudocode statement.

Line number .....

Explanation .....

..... [1]

6 Study the sequence of pseudocode statements.

```
CONST  a = 3.2   : REAL
DECLARE x, y, z, Answer1, Answer2, Answer3  : REAL
DECLARE p, q     : BOOLEAN
x ← 3
x ← x + 7
y ← 6
Answer1 ← 2 * (a + y)
z ← 6
Answer2 ← y ^ 2 + 5
p ← TRUE
q ← NOT(NOT(p))
Answer3 ← y + a * 2
```

Give the final value assigned to each variable.

(i) `x` ..... [1]

(ii) `Answer1` ..... [1]

(iii) `Answer2` ..... [1]

(iv) `q` ..... [1]

(v) `Answer3` ..... [1]

Appendix

Built-in functions (pseudocode)

```
ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
```

returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.

For example: `ONECHAR("New York", 5)` returns `'Y'`

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```

returns the number of characters in `ThisString`.

For example: `CHARACTERCOUNT("New York")` returns `8`

```
SUBSTR(ThisString : STRING, Value1 : INTEGER, Value2 : INTEGER) RETURNS
STRING
```

returns a sub-string from within `ThisString`.
`Value1` is the start index position (counting from the left, starting with 1).
`Value2` is the final index position.

For example: `SUBSTR("art nouveau", 5, 11)` returns `"nouveau"`

```
TONUM(ThisString : STRING) RETURNS INTEGER or REAL
```

returns the integer or real equivalent of the string `ThisString`.

For example: `TONUM("502")` returns the integer `502`
`TONUM("56.36")` returns the real number `56.36`

```
ASC(ThisCharacter : CHAR) RETURNS INTEGER
```

returns an integer which is the ASCII character code for the character `ThisCharacter`.

For example: `ASC('A')` returns integer `65`

© UCLES 2016

`CHR(Value : INTEGER) RETURNS CHAR`

returns the character that ASCII code number `Value` represents.

For example: `CHR(65)` returns `'A'`

`RND() RETURNS REAL`

returns a random number in the range 0 to 0.99999

For example: `RND()` returns `0.67351`

`INT(ThisNumber : REAL) RETURNS INTEGER`

returns the integer part of `ThisNumber`.

For example: `INT(12.79)` returns `12`

Errors

For any function, if the program calls the function incorrectly, the function returns an error.

Concatenation operator

`&` operator – Concatenates two expressions of `STRING` or `CHAR` data type.

For example: `"South" & " " & "Pole"` produces `"South Pole"`
`'B' & "000654"` produces `"B000654"`

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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

© UCLES 2016

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

8494377800

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 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 **20** pages. Any blank pages are indicated.

DC (RW) 206327/4
© UCLES 2021

1 **(a)** Design and coding are stages of the program development cycle.

State **two** activities from each of these stages.

Design stage

Activity 1 .....

Activity 2 .....

Coding stage

Activity 1 .....

Activity 2 .....

[4]

(b) The following is part of a procedure to manage the stock in a shop:

```
PROCEDURE InitVars()
   DECLARE Vs  : STRING  // program version
   DECLARE Pe  : REAL    // stock item purchase price
   DECLARE Exp : STRING  // date when item expires
   DECLARE S_lr: STRING  // supplier ID
   DECLARE Rp  : REAL    // low stock value
   DECLARE OUT : BOOLEAN // any items in stock ?
…
ENDPROCEDURE
```

(i) Give **two** reasons why the identifier names used in the procedure `InitVars()` are **not** examples of good practice.

Reason 1 .....

.....

Reason 2 .....

.....

[2]

(ii) Complete the following table by writing an appropriate identifier name for each of these four lines in the procedure `InitVars()`.

Line	Appropriate identifier name
102	
103	
105	
106	

[4]

(iii) Line `101` of the procedure needs to be changed because:

- the identifier should be declared as a fixed value that does not change during program execution
- the identifier name is not appropriate
- the program version should be `ver1.5.8`

Write the new correct statement for line `101`.

.....

..... [2]

(c) A program can store characters using either the ASCII or the Unicode character set.

Give **two** reasons why Unicode is preferred to the ASCII character set for storing characters.

Reason 1 .....

.....

.....

Reason 2 .....

.....

.....

[2]

2 A café manager needs a program for a loyalty scheme. A customer is awarded one loyalty point for each purchase.

When a customer has ten or more points, the customer is offered a free slice of cake.

(a) During the program design, a decision is made to store the loyalty data in a text file called `LOYALTY.txt`. This file holds the membership number and total number of points for each customer on a single line.

Part of the program will check whether a particular customer has enough points for a free slice of cake.

Describe the algorithm for this part of the program using **structured English**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) In another part of the program, the following statements assign values to variables.

```
DayOfMonth ← "15"
IsMember ← TRUE
DOB ← "22042001"
Firstname ← "Seanna"
Lastname ← "Adamson"
Points ← 12
```

Complete the table by writing the pseudocode expression that matches the description given **and** the value that each expression evaluates to.

Refer to the **Appendix** on pages 18–19 for a list of built-in pseudocode functions and operators.

Description of expression	Pseudocode expression	Evaluates to
Evaluates to `TRUE` if `DayOfMonth` is within the first seven days of the month		
Concatenates the second and third letters of `Firstname` with the last three letters of `Lastname`		
Evaluates to `TRUE` if `DOB` contains eight characters		
Evaluates to `TRUE` if the customer is a member and has enough points for a free slice of cake		

[4]

(c) Another part of the loyalty scheme program will need to access additional files. A filename contains a three-letter extension that represents its file type.

For example, the filename `"thisfile.txt"` has the extension `"txt"` and file type `"Text"`. The filename will always be in lower case.

A function `GetFileType()` returns a string containing the description of the file type.

The function uses selection constructs as follows:

```
FUNCTION GetFileType(Filename : STRING) RETURNS STRING
   DECLARE FileExt  : STRING
   DECLARE FileType : STRING

   FileExt ← RIGHT(Filename, 3)

   IF FileExt = "rtf"
     THEN
       FileType ← "Rich text format"
   ENDIF

   IF FileExt = "csv"
     THEN
       FileType ← "Comma separated values"
   ENDIF

   IF FileExt = "txt"
     THEN
       FileType ← "Text"
     ELSE
       FileType ← "Unknown"
   ENDIF

   RETURN FileExt
ENDFUNCTION
```

Rewrite the function as follows:

- Replace the `IF` structures with a `CASE` structure.
- Correct the logic error in the function.

Refer to the **Appendix** on pages 18–19 for a list of built-in pseudocode functions and operators.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

3 **(a)** Describe the term **decomposition** when used to develop algorithms.

.....

.....

.....

.....

.....

..... [3]

(b) Explain the term transferable skills **and** state how these skills are used in program development.

.....

.....

.....

..... [2]

4 A global 1D array `Flower` represents the flowers in a field and is defined as follows:

- The array contains 20 elements of type integer.
- Each element of the array contains a number associated with each flower.

(a) A procedure, `InitialiseArray()`, initialises the array `Flower` by setting the value of each element to –1.

Write **pseudocode** for the procedure `InitialiseArray()`.

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) A procedure called `RandomPath()` assigns values to the array `Flower` to store a path through the flowers in the fields as follows:

- the first flower to be visited is flower number 10
- randomly select the next flower to be visited between 1 and 20
- store the index of the next flower to be visited in the current element of the array `Flower` if the flower has not been visited
- visit each flower only once
- repeat until the path has been completed through all flowers.

An element of –1 indicates that a flower has not been visited.

Write **program code** for the procedure `RandomPath()`.

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) The procedure `RandomPath()` executes but gives an unexpected result.

(i) State the type of program error that has occurred **and** identify how this error can be detected.

Type of program error .....

Method of detection .....

.....
[2]

(ii) Assume that the random number has been incorrectly generated in the procedure `RandomPath()`.

State **two** possible consequences of the incorrect use of the random number function.

1 .....

.....

2 .....

.....
[2]

5 The pseudocode for a procedure `FormatYear()` is written as follows:

```
PROCEDURE FormatYear(Year : INTEGER, UseSuffix : BOOLEAN)

   DECLARE OutYear : STRING
   CONSTANT SUFFIX1 = "BCE"
   CONSTANT SUFFIX2 = "CE"

   OutYear ← NUM_TO_STRING(Year)

   IF UseSuffix = TRUE
     THEN
       IF Year < 0
         THEN
           OutYear ← OutYear & SUFFIX1
         ELSE
           OutYear ← OutYear & SUFFIX2
         ENDIF
   ENDIF
   OUTPUT OutYear
ENDPROCEDURE
```

(a) Draw a program flowchart to represent the algorithm for the module.

Variable and constant declarations are not required in program flowcharts.

[5]

(b) For each of the following tests, choose **three** values of the parameters `Year` and `UseSuffix` that test **three different** aspects of the procedure `FormatYear()`.

State the expected output in each case.

Test 1

Parameter	Value	Expected output
`Year`		
`UseSuffix`		

Test 2

Parameter	Value	Expected output
`Year`		
`UseSuffix`		

Test 3

Parameter	Value	Expected output
`Year`		
`UseSuffix`		

[6]

(c) **(i)** Describe **two** features of black-box testing.

.....

.....

.....

..... [2]

(ii) Describe **two** features of white-box testing.

.....

.....

.....

..... [2]

BLANK PAGE

6 A geocode string consists of four characters that are followed by:

- the character `'+'`
- two more characters
- a comma
- a description of the location.

For example:

- The geocode string for Cambridge, UK is `"646A+6R,CambridgeUK"`.
- The geocode string for Chicago, USA is `"V9PG+3P,ChicagoUSA"`.

A program uses two 1D arrays to store data about the locations in which photographs were taken.

- An array `GeoCodeData` contains a geocode string for each location. A geocode string is added to this array if it does not exist in the array.

- An array `GeoCodeLog` contains the first seven characters of the geocode string for each photograph, followed by a space and the date when the photograph was taken. The date format is `DD/MM/YYYY`.

  For example, a photograph taken in Cambridge, UK on 21/12/2020 is represented as:

  ```
  "646A+6R 21/12/2020"
  ```

Assume that:

- both arrays contain 20 000 elements
- new data is added to the next unused element
- unused array elements are indicated by the string `"AAAA+0A"`.

(a) A programmer has started to define the program module `SearchLog()`.

Module	Description
`SearchLog()`	• takes the first seven characters of a geocode as a parameter • counts the number of times this parameter occurs in the array `GeoCodeLog` • extracts the date the last photograph was taken at the geocode location • returns a string formed by concatenating the count, a single space and the date

Write **pseudocode** for the module `SearchLog()`.

.....

.....

.....

.....

.....

.....

..... [6]

(b) A text file, `Locations.txt`, stores data from the array `GeoCodeData` combined with data extracted from the array `GeoCodeLog`. Each line of the text file contains three items of data in the following format:

- The geocode stored in each element of the array `GeoCodeData` followed by a hash symbol (`#`).
- The number of times each geocode is found in the array `GeoCodeLog` followed by the hash symbol.
- The date of the last photograph taken at that location.

For example, twelve photographs were taken at geocode `V9PG+3P,ChicagoUSA`.
The last photograph was taken on 09/10/2020.

The corresponding entry in the file `Locations.txt` is:

```
"V9PG+3P,ChicagoUSA#12#09/10/2020"
```

The program module `ExtractArrays()` is defined as follows:

Module	Description
`ExtractArrays()`	• creates the text file `Locations.txt` • uses `SearchLog()` to obtain the string of the count and the date of the last photograph for each geocode location • writes the entry in the text file `Locations.txt`

You may assume that unused elements of `GeoCodeData` array are not added to the file `Locations.txt`.

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

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`

```
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`

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.

© UCLES 2021