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.

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.

.....

.....

.....

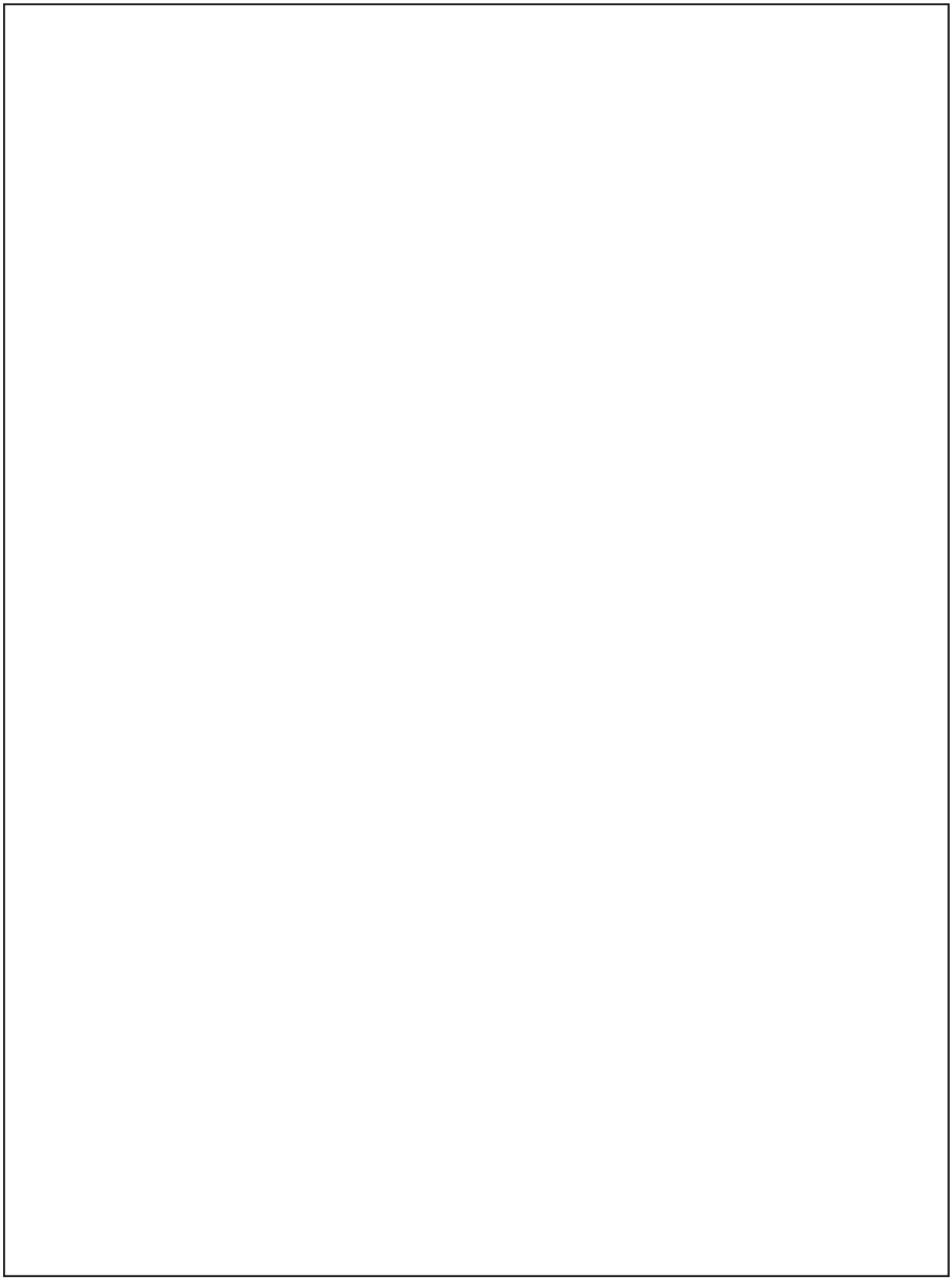
.....

.....

..... [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