Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

3263229235

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.

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

.....

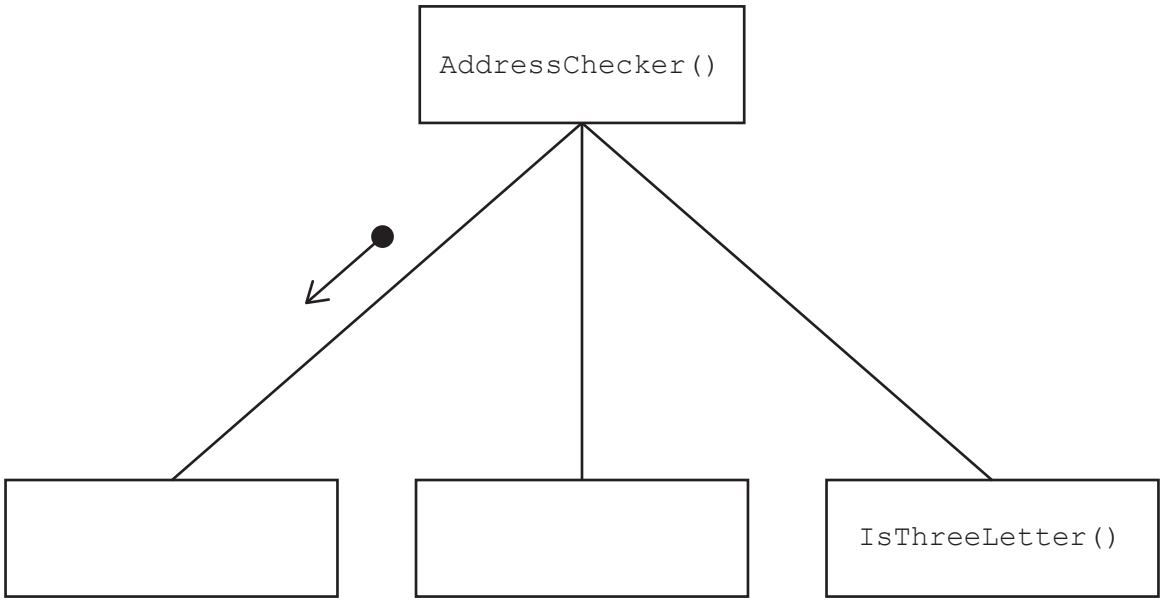
.....

.....

..... [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`.

```
01 DECLARE MatchItems : ARRAY[1:100] OF STRING

10 FUNCTION Extract(FileName : STRING, Match : STRING,
                    Signal : INTEGER) RETURNS BOOLEAN
11
12    DECLARE FileLine : INTEGER
13    DECLARE Counter : INTEGER
14    DECLARE FileData, SubMatch : STRING
15    DECLARE Result : BOOLEAN
16
17    Counter ← 0
18    FileLine ← 1
19    Result ← FALSE
20
21    OPENFILE FileName FOR READ
22
23    WHILE NOT EOF(FileName) AND FileLine <= 100
24       READFILE FileName, FileData
25       SubMatch ← LEFT(FileData, LENGTH(Match))
26       IF SubMatch = Match
27          THEN
28             MatchItems[Counter] ← FileData
29             Counter ← Counter + 1
30       ENDIF
31       FileLine ← FileLine + 1
32    ENDWHILE
33
34    CLOSEFILE FileName
35    IF Counter >= Signal
36       THEN
37         Result ← TRUE
38    ENDIF
39    RETURN Result
40
41 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.

```
23    WHILE NOT EOF(FileName) AND FileLine <= 100
24       READFILE FileName, FileData
25       SubMatch ← LEFT(FileData, LENGTH(Match))
26       IF SubMatch = Match
27          THEN
28             MatchItems[Counter] ← FileData
29             Counter ← Counter + 1
30       ENDIF
31       FileLine ← FileLine + 1
32    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