Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	
CENTRE NUMBER	
CANDIDATE NUMBER	

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.

[Turn over

Question 1 begins on the next page.

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)`	

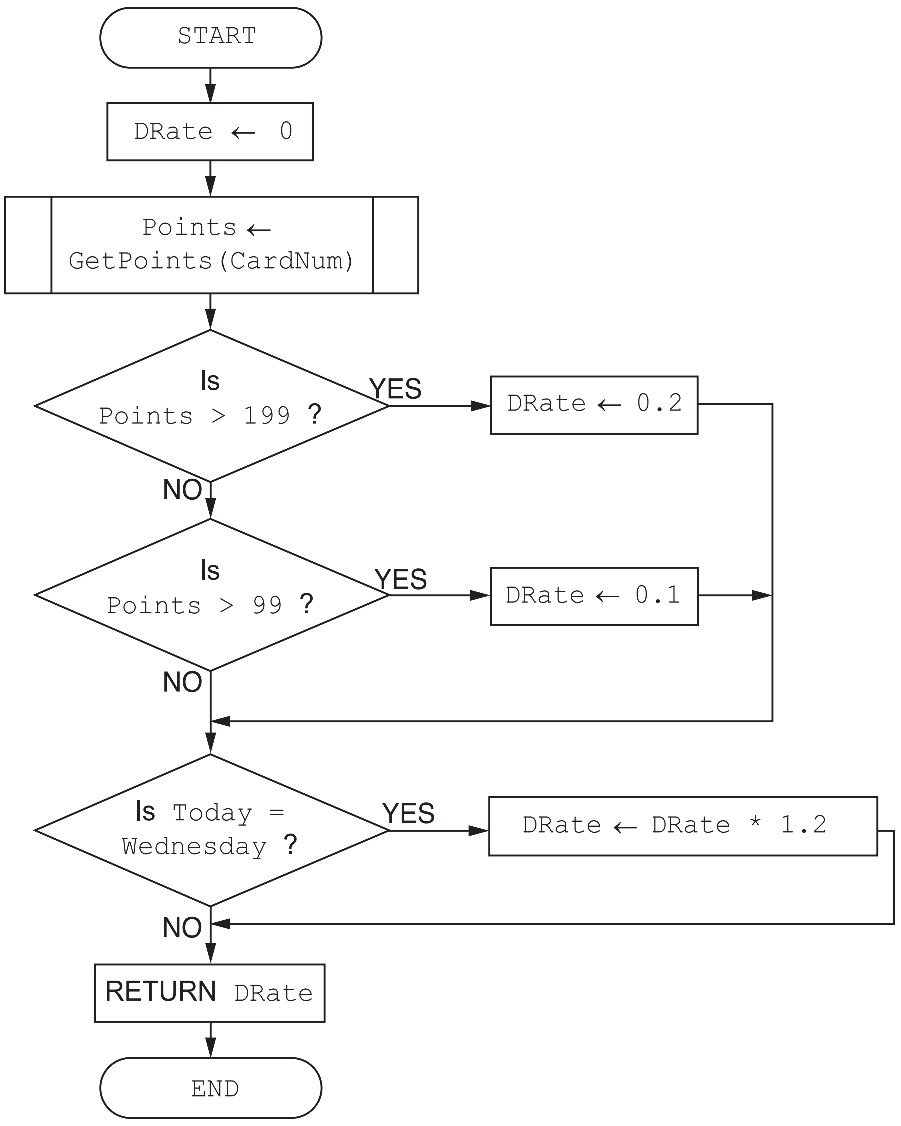
[5]

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`