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.

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.

```
10 FUNCTION Check(InString : STRING) RETURNS BOOLEAN
11
12   DECLARE Index : INTEGER
13   DECLARE StrLen : INTEGER
14   DECLARE NumUpper, NumLower : INTEGER
15   DECLARE NumNonAlpha : INTEGER
16   DECLARE NextChar : CHAR
17
18   NumUpper ← 0
19   NumLower ← 0
20   NumNonAlpha ← 0
21
22   StrLen ← LENGTH(InString)
23   IF StrLen < 7
24      THEN
25         RETURN FALSE
26      ELSE
27         FOR Index ← 1 TO StrLen
28            NextChar ← MID(InString, Index, 1)
29            IF NextChar >= 'a' AND NextChar <= 'z'
30               THEN
31                  NumLower ← NumLower + 1
32               ELSE
33                  IF NextChar > 'A' AND NextChar <= 'Z'
34                     THEN
35                        NumUpper ← NumUpper + 1
36                     ELSE
37                        NumNonAlpha ← NumNonAlpha + 1
38                  ENDIF
39            ENDIF
40         ENDFOR
41   ENDIF
42
43   IF (NumUpper >= 1) AND (NumLower >= 1) AND (NumNonAlpha >= 1)
44      THEN
45         RETURN TRUE
46      ELSE
47         RETURN FALSE
48   ENDIF
49
50 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.

.....

.....

.....

.....

.....

.....

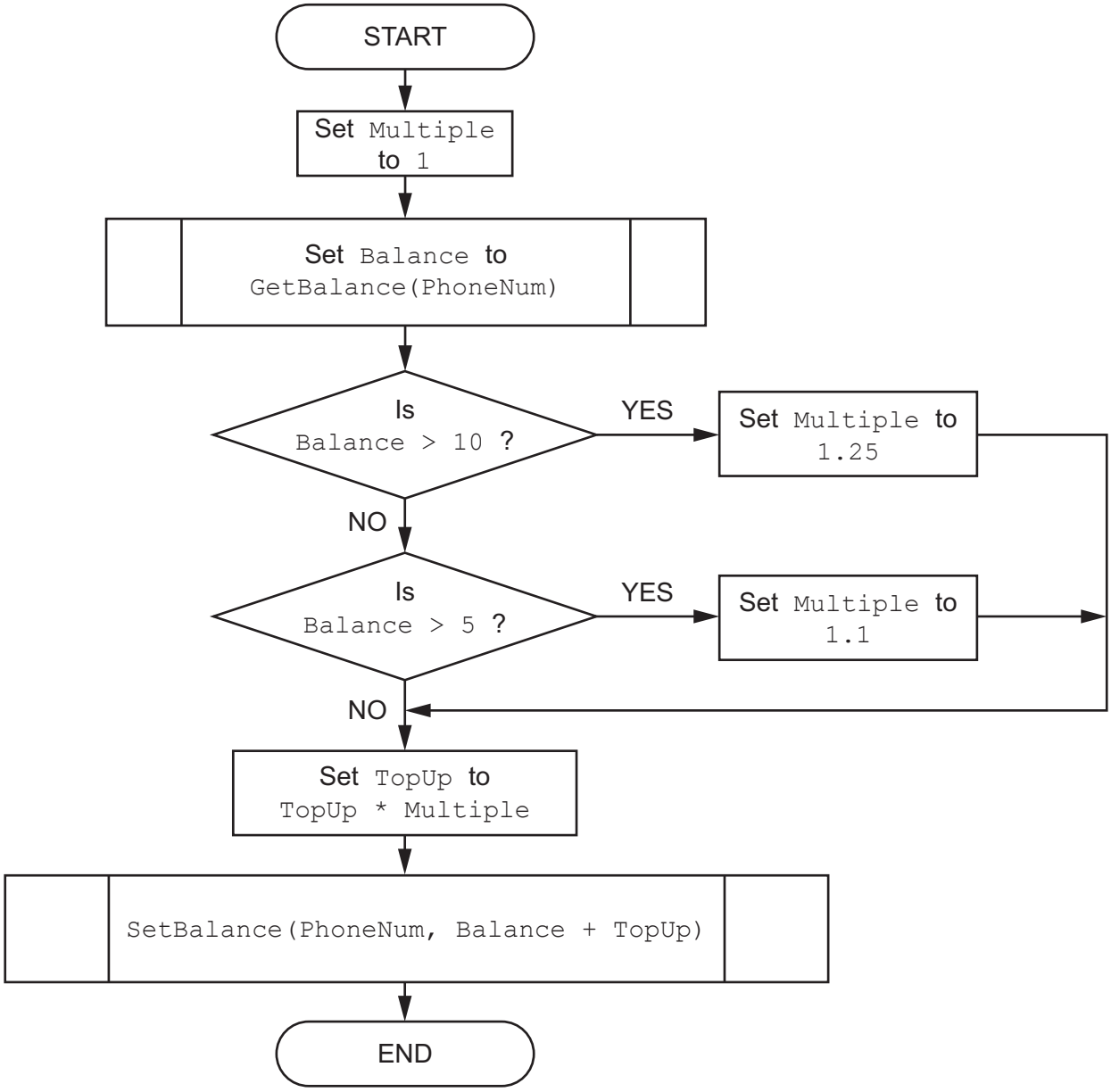
..... [4]

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:
	  o a `STRING` value representing a time
	  o 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:
	  o currently not on hire, or
	  o 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:
	  o an `INTEGER` value representing the boat number
	  o a `STRING` value representing the hire start time
	  o an `INTEGER` value representing the hire duration in minutes
	  o 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