Cambridge International AS & A Level

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 **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

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 .....

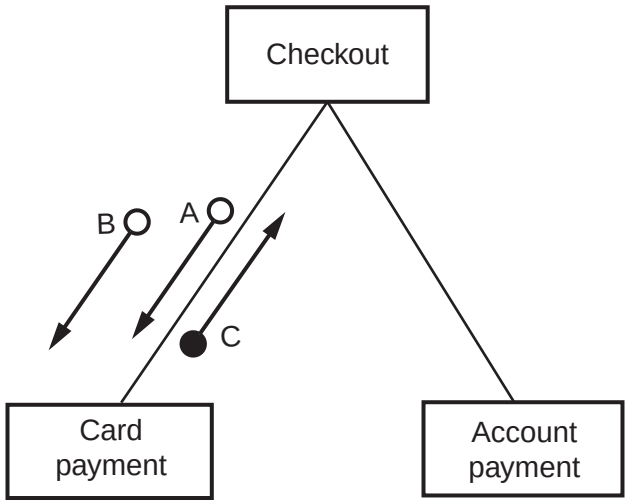
.....

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`**

BLANK PAGE

BLANK PAGE