Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME: **Sample answers**

CENTRE NUMBER | | | | | |

CANDIDATE NUMBER | | | | |

0123456789

COMPUTER SCIENCE **9618/02**

Paper 2 Fundamental Problem-solving and Programming Skills **For examination from 2021**

SPECIMEN PAPER **2 hours**

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

1 **(a)** Program variables have values as follows:

Variable	Value
Today	"Tuesday"
WeekNumber	37
Revision	'C'
MaxWeight	60.5
LastBatch	TRUE

(i) Give an appropriate data type for each variable.

Variable	Data type
Today	**STRING**
WeekNumber	**INTEGER**
Revision	**CHAR**
MaxWeight	**REAL**
LastBatch	**BOOLEAN**

[5]

(ii) Evaluate each expression in the following table.
If an expression is invalid then write ERROR.

Refer to the **Insert** for the list of pseudocode functions and operators.

Expression	Evaluates to
MID(Today, 3, 2) & Revision & "ape"	**"esCape"**
INT(MaxWeight + 4.2)	**64**
LENGTH(MaxWeight)	**ERROR**
MOD(WeekNumber, 12)	**1**
(Revision <= 'D') AND (NOT LastBatch)	**False**

[5]

(b) Simple algorithms usually consist of input, process and output.

Complete the table to show if each statement is an example of input, process or output.
Place one or more ticks (✓) for each statement.

Item	Statement	Input	Process	Output
1	SomeChars ← "Hello World"		√	
2	OUTPUT RIGHT(SomeChars, 5)		√	√
3	READFILE MyFile, MyChars	√	√	
4	WRITEFILE MyFile, "Data is " & MyChars		√	√

[4]

(c) Write in pseudocode a **post-condition loop** to output all the odd numbers between 100 and 200.

Method 1

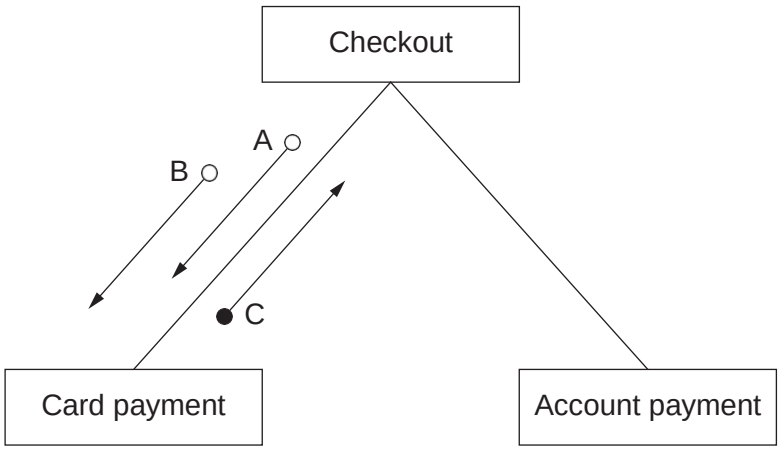
```
C←100
REPEAT
    IF (C MOD 2)=1        // Odd number
        OUTPUT C
    ENDIF
    C←C+1
UNTIL C=200
```

Method 2

```
C←101
REPEAT
    OUTPUT C
    C←C+2
UNTIL C=199
```

..... [4]

2 Roberta downloads music from an online music store. The diagram shows part of a structure chart for the online music store program.



(a) State **three** items of information that the diagram shows about the design of the program.

1 This diagram has identified the three modules – Checkout, Card payment and Account payment

2 The hierarchy of the modules is shown

3 It also shows the transfer of data between modules through parameters

[3]

(b) Examples of the data items that correspond to the arrows are given in the table.

Arrow	Data item
A	`234.56`
B	`"Ms Roberta Smith"`
C	`TRUE`

Use pseudocode to write the function header for the Card payment module.

```
FUNCTION CardPayment(Amount : REAL, Name : STRING) RETURNS BOOLEAN
```

..... [3]

3 A stack is created using a high-level language. The following diagram represents the current state of the stack. The Top of Stack pointer points to the last item added to the stack.

Address	Value	Pointer
99		
100		
101	E	⇐ TopOfStack
102	D	
103	C	
104	B	
105	A	

a) Two operations associated with this stack are `PUSH()` and `POP()`.

Describe these operations with reference to the diagram.

```
MyVar = POP()
```

Variable MyVar will be assigned the value 'E' which will be removed from the stack. TopOfStack will increase to 102

.....

```
PUSH('Z')
```

'Z' will be stored in address 101
TopOfStack will decrease to 101

.....

[4]

b) Two programs use a stack to exchange data. Program `AddString` pushes a string of characters onto the stack one character at a time. Program `RemoveString` pops the same number of characters off the stack, one character at a time. The string taken off the stack is different from the string put on the stack.

Explain why the strings are different.

Stack works according to LIFO method. The last character of the string will be the first character of the new string when popped by RemoveString

..... [2]

4 **(a)** Parameter `x` is used to pass data to procedure `MyProc` in the following pseudocode:

```
x ← 4
CALL MyProc(x)
OUTPUT x
```

```
PROCEDURE MyProc(x : INTEGER)
    DECLARE z : INTEGER
    x ← x + 1
    z ← x + 3
ENDPROCEDURE
```

There are two parameter passing methods that could be used.

Complete the following table for each of the two methods.

Name of parameter passing method	Value output	Explanation
By reference	5	.The address of the variable is passed so the variable gets changed when parameter values change within the module ....
By value	4	A copy of the value is passed without any reference to the original variable. So the original variable value remains the same, even when values change within the module. .....

[6]

(b) The pseudocode includes the use of parameters.

State **two** other features in the pseudocode that support a modular approach to programming.

1 .It shows procedures and

.....

2 Local variables which support modular programming

.....

[2]

5 A company keeps details of its product items in a 1D array, `Stock`. The array consists of 1000 elements of type `StockItem`.

The record fields of `StockItem` are:

Field	Typical value
`ProductCode`	`"BGR24-C"`
`Price`	`102.76`
`NumberInStock`	`15`

(a) Write pseudocode to declare the record structure `StockItem`.

```
TYPE StockItem
    DECLARE ProductCode: String
    DECLARE Price: Real
    DECLARE NumberInStock: Integer
ENDTYPE
```

..... [3]

(b) Write pseudocode to declare the `Stock` array.

```
DECLARE Stock : ARRAY[1:1000] of StockItem
```

..... [3]

(c) Write pseudocode to modify the values to element 20 as follows:

- set the price to 105.99
- increase the number in stock by 12

```
Stock[20].Price ← 105.99
Stock[20].NumberInStock ← Stock[20].NumberInStock + 12
```

..... [2]

(d) A stock report program is developed.

Write pseudocode to output the information for each stock item that has a price of at least 100.

Output the information as follows:

Product Code: BGR24-C Number in Stock: 15

```
DECLARE n : INTEGER
FOR n ← 1 TO 1000
        IF Stock[n].Price >= 100
        THEN
                OUTPUT "ProductCode: " & Stock[n].ProductCode &
                " Number in Stock: " & Stock[n].NumberInStock
        ENDIF
NEXT
```

..... [4]

6 Members of a family use the same laptop computer. Each family member has their own password. To be valid, a password must comply with the following rules:

1 At least two lower-case alphabetic characters
2 At least two upper-case alphabetic characters
3 At least three numeric characters
4 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 it is a valid password
- FALSE otherwise

(a) Write pseudocode to implement the function `ValidatePassword`.
Refer to the **Insert** for the list of pseudocode functions and operators.

```
FUNCTION ValidatePassword(Pass : STRING) RETURNS BOOLEAN
        DECLARE LCaseChar, UCaseChar, NumChar, n : INTEGER
        DECLARE NextChar : CHAR
        DECLARE ReturnFlag : BOOLEAN
        ReturnFlag ← TRUE
        LCaseChar ←  0
        UCaseChar ←  0
        NumChar ← 0
        n ← 1
        WHILE n <= LENGTH(Pass) AND ReturnFlag = TRUE
                NextChar ← MID(Pass, n, 1)
                IF NextChar >= 'a' AND NextChar <= 'z'
                THEN
                        LCaseChar ← LCaseChar + 1
                ELSE
                        IF NextChar >= 'A' AND NextChar <= 'Z'
                        THEN
                                UCaseChar ← UCaseChar + 1
                        ELSE
                                IF NextChar >= '0' AND NextChar <= '9'
                                THEN
                                        NumChar ← NumChar + 1
                                ELSE
                                        ReturnFlag ← FALSE      //illegal character
                                ENDIF
                        ENDIF
                ENDIF
                n←n+1
        ENDWHILE
    IF LCaseChar > 1 AND UCaseChar > 1 AND NumChar > 2 AND ReturnFlag
    THEN
            ReturnFlag ← TRUE

    ELSE
            ReturnFlag ← FALSE
    ENDIF
    RETURN ReturnFlag
ENDFUNCTION
```

..... [9]

(b) The `ValidatePassword` function will be tested.

(i) Give a valid password that can be used to check that the function returns TRUE under the correct conditions.

Password1: "**PQab321"** ..... [1]

(ii) Password1 is modified to test each rule separately. Give **four** modified passwords and justify your choice.

Password to test rule 1: **"Pab321"**

Reason: **To test Uppercase characters**

Password to test rule 2: **"PQb321"**

Reason: **To test lowercase characters**

Password to test rule 3: **"PQab32"**

Reason: **To test numeric characters**

Password to test rule 4: **"PQab*321"**

Reason: **To test invalid characters**

[4]

(iii) When testing the `ValidatePassword` function a module it is necessary to test all possible paths through the code.

State the name given to this type of validation testing.

This kind of validation is called Whitebox testing [1]

(iv) A program consisting of several functions can be tested using a process known as 'stub testing'

Explain this process.

- **It is useful to carry out testing even before the modules are completed**
- **Module stubs can provide a known response to confirm that a call to the module will really work**.

..... [2]

7 `LogArray` is a 1D array containing 500 elements of type `STRING`.

A procedure, `LogEvents`, is required to add data from the array to the end of the existing text file `LoginFile.txt`

Unused array elements are assigned the value `"Empty"`. These can occur anywhere in the array and should **not** be added to the file.

Write pseudocode for the procedure `LogEvents`.

Refer to the **Insert** for the list of pseudocode functions and operators.

```
PROCEDURE LogEvents()
    DECLARE FileData : STRING
    DECLARE ArrayIndex : INTEGER
    OPENFILE "LoginFile.txt" FOR APPEND
    FOR ArrayIndex ← 1 TO 500
        IF LogArray[ArrayIndex]<> "Empty"
        THEN
            FileData ← LogArray[ArrayIndex]
            WRITEFILE "LoginFile.txt", FileData
        ENDIF
    NEXT
    CLOSEFILE "LoginFile.txt"
ENDPROCEDURE
```

..... [8]