

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME	MUHAMMAD BURHAN KHAN
CENTRE NUMBER	C R E D O
CANDIDATE NUMBER	





COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2025**

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.

This document has **24** pages. Any blank pages are indicated.

© UCLES 2025

CS BY AQIB KHAN

THE AQIB KHAN AN INSTITUTE OF O/A LEVEL

TAK (The AQIB KHAN)- An institute of O/A level

Aki.Khan CS By AQIB Khan CS By AQIB KHAN TheAkikhan.com +92 3145030717

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

1 A user requires a program that implements complex data encryption to a recognised international standard. A programmer has started the design of the program.

(a) The programmer decides to use a library routine to provide part of the solution. One reason for this is that library routines may perform functions that the programmer is unable to program themselves.

State **three** other benefits of using library routines in the development of the program.

1 Tried and tested

2 Readily available.

3 They will be updated automatically when improvements are made.

[3]

(b) The programmer has identified a different part of the algorithm that would be appropriate to implement as a subroutine (a procedure or a function); no suitable library routine exists for this part.

(i) State **two** reasons why the programmer may decide to use a subroutine.

1 It can be called multiple times from different parts of the main program without rewriting the code, saving time.

2 As a large, complex problem can be broken down into smaller sub-modules, makes the code easier to debug and maintain.

[2]

(ii) A function header in pseudocode is defined as:

```
FUNCTION Pass2(Count : INTEGER) RETURNS BOOLEAN
```

Complete the table by describing the terms used in the function header.

The first row has been completed.

Term	Description
`Pass2`	the name of the function
`Count`	parameter/value passed to function
`BOOLEAN`	data type returned by function

[2]

© UCLES 2025

TAK (The AQIB KHAN) An institute of O/A level
Aki.Khan CS By AQIB Khan CS By AQIB KHAN TheAkikhan.com +92 3145030717
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

(c) Variables in the program have example values:

Variable	Example value
`DoB`	`23/6/2011`
`Multiplier`	`2.5`
`AddressLine[1]`	`"35 Lincoln Avenue"`

Complete the table by evaluating each expression using the example values:

Expression	Evaluates to
`LENGTH(NUM_TO_STR(Multiplier))` LENGTH("2.5")	3
`MONTH(DoB) > 4` 6 > 4	TRUE
`15 + STR_TO_NUM(MID(AddressLine[1], 2, 1))` "5" 15 + 5 = 20	20

[3]

CS BY AQIB KHAN
THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

TAK (The AQIB KHAN) An institute of O/A level
Aki.Khan CS By AQIB Khan CS By AQIB KHAN TheAkikhan.com +92 3145030717

2 **(a)** A program contains a global 1D array of type `STRING` containing 65 elements.

An existing text file is used to store the data in the array. Only non-blank elements (those that do **not** contain an empty string) are written to the text file.

An algorithm will:

- write the first element of the array as a new line in the text file
- continue until all elements have been written, each to a new line of the text file.

Stepwise refinement is applied to the algorithm.

Describe up to **six** steps for this algorithm that could be used to produce pseudocode.

Do **not** use pseudocode statements in your answer.

Step 1 Open the existing text file for writing.

Step 2 Set a counter to point to the first element of the array.

Step 3 Check whether the current array element is not an empty string.

Step 4 If the element is not empty, write it to the text file on a new line.

Step 5 Move to the next element and repeat the check until all the 65 elements have been processed.

Step 6 Close the text file.

[6]

(b) Iteration is one programming construct.

Identify **one** other programming construct that will be required when the algorithm from part **(a)** is converted into pseudocode and explain how it is used.

Construct Selection.

Use Check if an array element is blank.

[2]

DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN
© UCLES 2025
TAK (The AQIB KHAN) An institute of O/A level
Aki.Khan CS By AQIB Khan CS By AQIB KHAN TheAkikhan.com +92 3145030717

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN
THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

3 A program uses a stack to hold up to 30 integer values. The stack is implemented using a global integer variable and a global 1D array.

The array is declared in pseudocode:

```
DECLARE ThisStack : ARRAY[1:30] OF INTEGER
```

Stack design notes:

- The global variable `SP` acts as a stack pointer. `SP` contains the array index of the last value pushed onto the stack.
- If the stack is empty, then `SP` is assigned the value zero.
- The first item added to the stack will be stored in `ThisStack[1]`
- `SP` is incremented each time an item is added to the stack.

A function `Pop()` is written to remove an item from `ThisStack`. The function returns an item of type `PopData` which is defined in pseudocode:

```
TYPE PopData
   DECLARE Data   : INTEGER
   DECLARE Exists : BOOLEAN
ENDTYPE
```

The value removed from the stack is assigned to `Data` and `Exists` is set to `TRUE`. If it is **not** possible to remove a value from the stack, then `Exists` is set to `FALSE`

Complete the pseudocode for `Pop()`

```
FUNCTION Pop() RETURNS PopData

   DECLARE ThisPop : .....PopData.....

   IF .....SP = 0..... THEN

      PopData.Exists ← .....FALSE..... // Stack is empty

   ELSE

      PopData.Data ← .....ThisStack[SP].....

      PopData.Exists ← .....TRUE.....

      SP ← .....SP – 1.....

   ENDIF

   RETURN ThisPop

ENDFUNCTION
```

[5]

DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN
THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

4 A program contains a global 1D array `Data` containing 20 elements of type `INTEGER`

A global string `NumString` represents a sequence of three-digit numbers, separated by commas. For example:

```
"101,456,219,754,328"
```

The string contains at least **four** three-digit numbers. The total number of three-digit numbers in the string is unknown.

A procedure `Store()` will:

- extract one three-digit number at a time from `NumString`
- convert each of the three-digit numbers extracted to an integer and assign this to the next array element, starting from index 1
- end when all three-digit numbers have been stored, or when the array is full.

Complete the pseudocode for `Store()`

All local variables used must be declared.

```
PROCEDURE Store()
    DECLARE Index: INTEGER
    DECLARE StartPos: INTEGER
    DECLARE TempNum: INTEGER

    Index <- 1
    StartPos <- 1

    // Loop continues until array is full or string is exhausted
    WHILE Index <= 20 AND StartPos <= LENGTH(NumString)

        // Extract 3 characters starting at StartPos
        TempNum <- MID(NumString, StartPos, 3)
        Data[Index] <- STR_TO_NUM(TempNum)
        Index <- Index + 1
        StartPos <- StartPos + 4
    ENDWHILE

ENDPROCEDURE
```

[6]

© UCLES 2025 9618/23/A/M/25

TAK (The AQIB KHAN) An institute of O/A level
Aki.Khan CS By AQIB Khan CS By AQIB KHAN TheAkikhan.com +92 3145030717
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN
THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

5 A global 1D array `Num` of integers contains four elements. A program assigns values to these elements as shown:

```
Num[1] ← 1
Num[2] ← 2
Num[3] ← 5
Num[4] ← 3
```

A procedure `Process()` manipulates the values in the array.

The procedure is written in pseudocode:

```
PROCEDURE Process(Start : INTEGER)
   DECLARE CaseVar, Index, Count : INTEGER

   Index ← Start
   Count ← 0

   WHILE Count <= 20
      CaseVar ← Num[Index]
      CASE OF CaseVar
         1 : Num[Index] ← Num[Index] + Index //clause one
             Index ← Index + 1
             Count ← Count + 1

         2 : Num[Index] ← Num[Index] + Index //clause two
             Index ← Index + 2
             Count ← Count + 2

         3 : Num[Index] ← Num[Index] * 2     //clause three
             Index ← Index + 3
             Count ← Count - 1

         4 : Count ← Count + 4               //clause four
         OTHERWISE : Count ← 20
      ENDCASE

      Index ← (Index MOD 4) + 1

   ENDWHILE

ENDPROCEDURE
```

THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL
CS BY AQIB KHAN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

(a) Complete the trace table by dry running the procedure when it is called in the statement:
`CALL Process(1)`

Index	CaseVar	Count	Num[1]	Num[2]	Num[3]	Num[4]

[5]

CS BY AQIB KHAN
THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

TAK (The AQIB KHAN) An institute of O/A level
Aki.Khan CS By AQIB Khan CS By AQIB KHAN TheAkikhan.com +92 3145030717

(b) As a reminder, the CASE structure in the pseudocode is:

```
CASE OF CaseVar
   1 : Num[Index] ← Num[Index] + Index //clause one
       Index ← Index + 1
       Count ← Count + 1

   2 : Num[Index] ← Num[Index] + Index //clause two
       Index ← Index + 2
       Count ← Count + 2

   3 : Num[Index] ← Num[Index] * 2      //clause three
       Index ← Index + 3
       Count ← Count - 1

   4 : Count ← Count + 4                //clause four
   OTHERWISE : Count ← 20
ENDCASE
```

The CASE structure could be optimised by combining two existing clauses.

(i) Identify the CaseVar values for these **two** existing clauses.

1 and 2. ..... [1]

(ii) Write pseudocode for a single clause to replace the **two** clauses identified in **(b) (i)**.

1 TO 2: Num[Index] <- Num[Index] + Index
Index <- Index + CaseVar
Count <- Count + CaseVar ..... [2]

THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN
THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

6 Students are learning about a simple check digit method for data validation. In this method, a single check digit is appended to the end of an original number to give a new number.

The students are studying a method which:

- calculates the sum of all the digits in the original number
- uses integer division to calculate the remainder when the sum is divided by 10
- uses the remainder as the check digit
- appends the check digit to the original number, creating the new number.

For example:

original number	4162
sum of all digits	4 + 1 + 6 + 2 = 13
remainder when the sum is divided by 10 using integer division	3
new number	41623

The original number is always at least three digits in length.

When the new number is input, the check digit is used to validate the new number.

(a) A function `Generate()` is written to take an original number as a parameter and to return the check digit.

Outline a test plan that could be used to fully test function `Generate()`

Assume that the parameter is valid.

Create test numbers that will produce each possible check digit value from 0 to 9. For each number, calculate expected remainder when the sum of digits is divided by 10 and check that the function returns the correct value. [2]

THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

(b) A function `CheckNumber()` will take an integer value and return the Boolean value `TRUE` if the check digit is correct, otherwise return `FALSE`

Write pseudocode for the function `CheckNumber()`

Assume that the parameter is valid.

```
FUNCTION CheckNumber (Number: INTEGER) RETURNS BOOLEAN
    DECLARE CheckDigit, OriginalNumber, Sum, Remainder: INTEGER

    CheckDigit <- Number MOD 10       // Get last digit
    OriginalNumber <- Number DIV 10   // check digit separated
    Sum <- 0

    WHILE OriginalNumber > 0
        Sum <- Sum + (OriginalNumber MOD 10)    //taking last digit and add
        OriginalNumber <- OriginalNumber DIV 10
    ENDWHILE

    Remainder <- Sum MOD 10       // calculated check digit

    IF Remainder = CheckDigit THEN
        RETURN TRUE
    ELSE
        RETURN FALSE
    ENDIF
ENDFUNCTION
```

[7]

CS BY AQIB KHAN
THE AQIB KHAN AN INSTITUTE OF O/A LEVEL

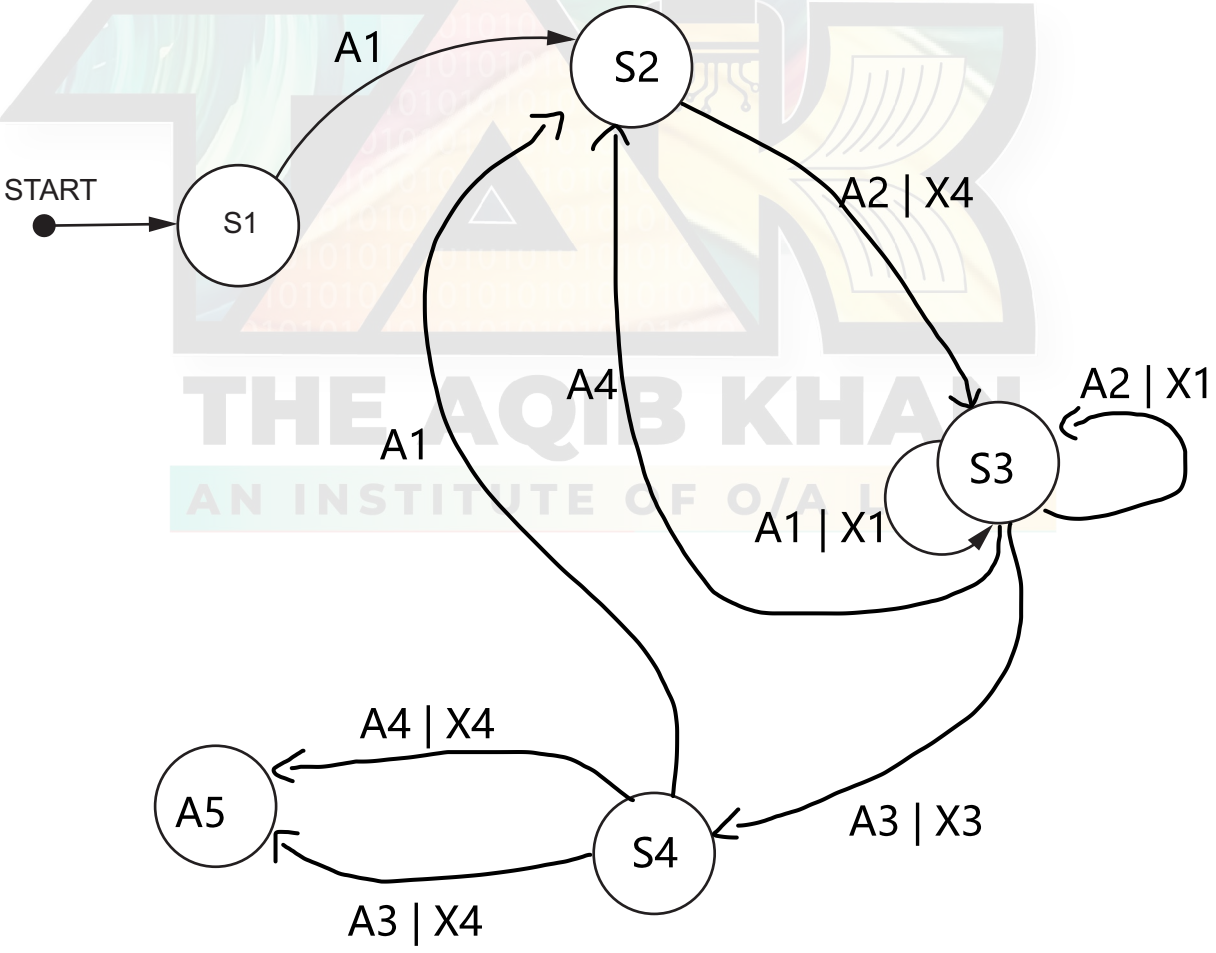
7 There are several different ways to express an algorithm during the design of a program.

(a) One part of the program contains an algorithm which is represented by a state-transition diagram.

The table shows the inputs, outputs and states for the algorithm:

Current state	Input	Output	Next state
S1	A1		S2
S2	A2	X4	S3
S3	A1	X1	S3
S3	A2	X1	S3
S3	A3	X3	S4
S3	A4		S2
S4	A3	X4	S5
S4	A4	X4	S5
S4	A1		S2

Complete the state-transition diagram to represent the information given in the table:



[5]

© UCLES 2025

TAK (The AQIB KHAN) An institute of O/A level
Aki.Khan CS By AQIB Khan CS By AQIB KHAN TheAkikhan.com +92 3145030717
CS BY AQIB KHAN
DO NOT WRITE IN THIS MARGIN

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

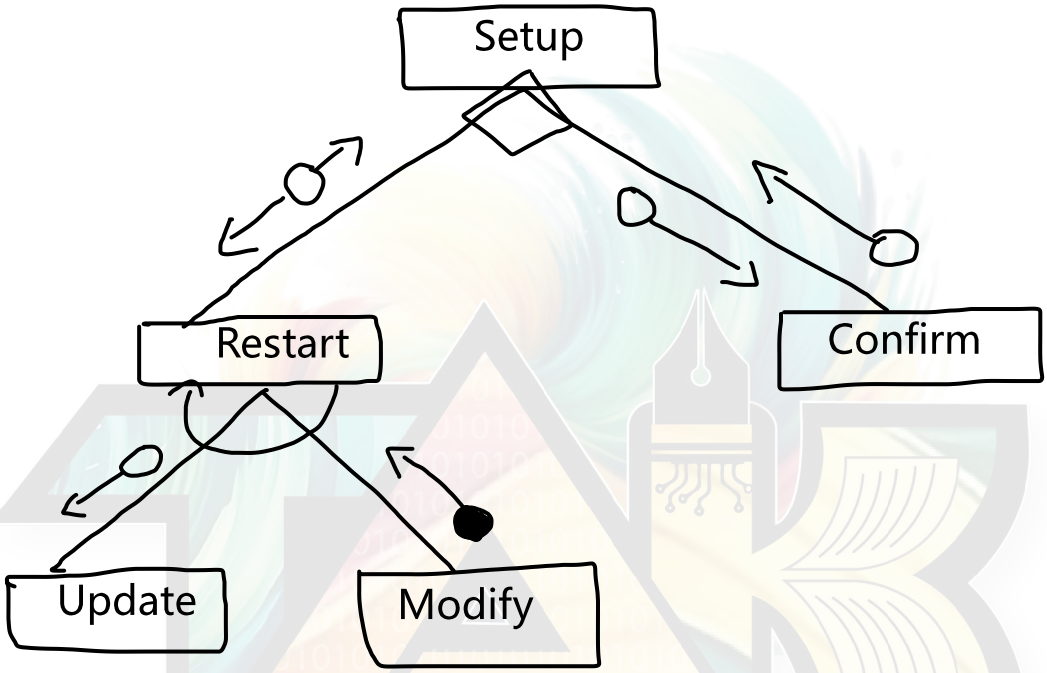
CS BY AQIB KHAN

(b) A structure chart is used to document a different part of the program, made up of five modules.

Program notes:

- module Setup calls either module Restart, or module Confirm
- module Confirm takes a string as a parameter and returns an integer
- module Modify has no parameters and returns a Boolean
- module Update takes a string as a parameter
- module Restart repeatedly calls module Update followed by module Modify
- module Restart takes a string as a parameter that is passed by reference.

Draw a structure chart to represent the relationship between the **five** modules, including all parameters and return values.



THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

[4]

TAK (The AQIB KHAN) An institute of O/A level
Aki.Khan CS By AQIB Khan CS By AQIB KHAN TheAkikhan.com +92 3145030717

8 A program is being developed to manage student book loans from a college library.

The programmer has defined a record type to define each loan.

The data items are:

Data item	Data type	Comment
`StudentID`	`STRING`	the unique ID of the student who has borrowed the book The first three characters of a `StudentID` represent a tutor ID. Each student has one tutor.
`BookID`	`STRING`	the unique ID of the book being borrowed
`OnLoan`	`BOOLEAN`	`TRUE` if the book has **not** been returned

The programmer has defined a global array `Loan` to store 8000 loan records.

There are more elements in the array than books in the library. Unused elements have the `StudentID` set to an empty string. These may occur anywhere in the array.

The programmer has defined a program module:

Module	Description
`LoanStatus()`	• called with two parameters of type `STRING` representing a `StudentID` and a `BookID` • outputs a message saying whether a given loan has been returned or not • outputs a warning message if a record of the given loan is **not** found

THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

(a) Write efficient pseudocode for the module `LoanStatus()`

Assume that each combination of `StudentID` and `BookID` can occur only once.

```
PROCEDURE LoanStatus(StudentID1, BookID1: STRING)
    DECLARE Index: INTEGER
    DECLARE Found: BOOLEAN

    Index <- 1
    Found <- FALSE

    WHILE Index <= 8000 AND Found = FALSE
        IF Loan[Index].StudentID <> " " THEN
            IF Loan[Index].StudentID = StudentID1 AND
               Loan[Index].BookID = BookID1 THEN
                IF Loan[Index].OnLoan = TRUE THEN
                    OUTPUT "Book has not been returned"
                ELSE
                    OUTPUT "Book has been returned"
                ENDIF
                Found <- TRUE
            ENDIF
        ENDIF

        Index <- Index + 1
    ENDWHILE

    IF Found = FALSE THEN
        OUTPUT "Loan record was not found"
    ENDIF
ENDPROCEDURE
```

[8]

CS BY AQIB KHAN
THE AQIB KHAN INSTITUTE OF O/A LEVEL

(b) A second module is defined:

Module	Description
`LoansPerTutor()`	• called with a parameter of type `STRING` representing a tutor ID (as a reminder, the first three characters of a `StudentID` represent a tutor ID) • returns an integer value representing the number of books currently on loan to students who have the given tutor

Reminder: unused elements have the `StudentID` set to an empty string. These may occur anywhere in the array.

Write pseudocode for the module `LoansPerTutor()`

```
FUNCTION LoansPerTutor(TutorID: STRING) RETURNS INTEGER
    DECLARE Index, Count: INTEGER
    DECLARE StudentTutorID: STRING

    Count <- 0

    FOR Index <- 1 TO 8000
        IF Loan[Index].StudentID <> " " THEN
            StudentTutorID <- LEFT(Loan[Index].StudentID, 3)  // Extract first 3 characters
            IF StudentTutorID = LoanTutor AND
                Loan[Index].OnLoan = TRUE THEN
                    Count <- Count + 1
            ENDIF
        ENDIF
    NEXT Index

    RETURN Count
ENDFUNCTION
```

[7]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

CS BY AQIB KHAN

(c) It is decided to mark as unused all records for book loans that have been returned. The data for these records will first be written to a new text file for archive purposes.

(i) The archive program will automatically generate a meaningful filename each time it is run.

Outline a meaningful format to use for the filename.

The filename could include a description such as "ReturnedLoans" followed by the current date (and time as well) in the format YYYYMMDD to ensure each file is unique.

..... [2]

(ii) There is a problem that will need to be overcome before the data items can be written to a text file.

As a reminder, the data items are:

Data item	Data type	Comment
`StudentID`	`STRING`	the unique ID of the student who has borrowed the book The first three characters of a `StudentID` represent a tutor ID. Each student has one tutor.
`BookID`	`STRING`	the unique ID of the book being borrowed
`OnLoan`	`BOOLEAN`	`TRUE` if the book has **not** been returned

Explain the problem.

OnLoan is not a string.

..... [1]

(iii) One way of storing the data items in a text file is to store each data item on a separate line.

Identify **one** benefit and **one** drawback of this way of storing the data.

Benefit Extracting a record is easier

.....

Drawback More file accesses needed (to read/write a complete record.)

.....

[2]

THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN
THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
CS BY AQIB KHAN

THE AQIB KHAN
AN INSTITUTE OF O/A LEVEL

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge Assessment International Education Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cambridgeinternational.org after the live examination series.

Cambridge Assessment International Education is part of Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.