Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME			
CENTRE NUMBER	□□□□□	CANDIDATE NUMBER	□□□□

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2024**

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 **20** pages. Any blank pages are indicated.

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

1 A program will calculate the tax payable based on the cost of an item.

Calculations will occur at many places in the program and these involve the use of one of three tax rates.

Tax rate values represent a percentage. For example, a tax rate value of 5.23 represents 5.23%. In this case, the tax payable on an item costing $100 would be $5.23.

Tax rate values are used at several places within the program. One example is given in pseudocode as follows:

```
HighRate ← FALSE
CASE OF ItemCost
   <= 50  : TaxRate ← 3.75      // tax rate of 3.75%
   <= 200 : TaxRate ← 5.23      // tax rate of 5.23%
   > 200  : TaxRate ← 6.25      // tax rate of 6.25%
            HighRate ← TRUE
ENDCASE
TaxPayable ← ItemCost * TaxRate // tax payable
```

(a) The pseudocode contains a logical error.

Identify the error **and** suggest a correction.

Error ....

....

Correction ....

.... [2]

(b) During the design of the program, tax rate values have been used wherever they are needed as shown in the pseudocode example above. Tax rates do not change while the program runs.

(i) Identify a more appropriate way of representing the tax rate values in the final program.

.... [1]

(ii) Describe the benefits of your answer to part **(b)(i)** with reference to this program.

....

....

....

....

....

.... [3]

(c) Give the **appropriate** data type for the variables in the following table, as used in the pseudocode:

Variable name	**Data type**
`HighRate`	
`TaxPayable`	

[2]

(d) The final `CASE` condition (`> 200`) in the pseudocode example could be replaced with a keyword.

Give the keyword.

.... [1]

2 A program uses three global integer variables `HH`, `MM` and `SS` to represent the current time in hours, minutes and seconds using the 24-hour clock notation.

Midnight would be represented as 00:00:00 (`HH`:`MM`:`SS`). If the variables `HH`, `MM` and `SS` contained the values 16, 30 and 10 respectively, then the time would be 16:30:10 or just after 4.30 in the afternoon.

A procedure `Tick()` will be called every second.
The procedure `Tick()` will:

- update the value in `SS` each time it is called
- update the values in `HH` and `MM` as appropriate
- call a procedure `CheckAlarm()` at the start of each minute
- call a procedure `NewDay()` whenever the time reaches midnight.

Complete the pseudocode for procedure `Tick()`.

```
PROCEDURE Tick()

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

..................................................................................

ENDPROCEDURE
```

[6]

BLANK PAGE

3 An algorithm will output the **last** three lines from a text file `Result.txt`

The lines need to be output in the same order as they appear in the file.

Assume:

- Three variables `LineX`, `LineY` and `LineZ` will store the three lines. These are of type string and all three variables have been initialised to an empty string.
- The file exists and contains **at least** three lines.

(a) The algorithm to output the lines is expressed in eight steps.

Complete the steps.

1. Open the file ....
2. Loop until ....
3. .... and store in `ThisLine`
4. Assign `LineY` to `LineX`
5. Assign `LineZ` to `LineY`
6. Assign `ThisLine` to `LineZ`
7. After the loop, ...
8. Output `LineX`, `LineY`, `LineZ`

[4]

(b) Explain the purpose of steps 4, 5 and 6 in the algorithm from part **(a)**.

....

....

.... [1]

(c) The requirement changes, and the algorithm will now output three lines from the file, starting from a **given** line number.

The modified algorithm will be implemented as a function which will:

- be called with an integer parameter representing the given line number
- output three lines, starting at the given line
- return `TRUE` if the 3 lines are output, or `FALSE` if it was not possible to output the 3 lines.

Describe the changes that need to be made to **steps 2 to 8** of the algorithm given in part **(a)**.

....

....

....

....

....

....

....

....

....

.... [4]

4 A program includes the following assignment statement:

```
Result ← STR_TO_NUM(x) / STR_TO_NUM(y)
```

When the program evaluates the expression in the statement, it performs a calculation.

Variable `Result` is of type real and variables `x` and `y` are of type string.

Two checks are required before the calculation is performed:

1. The two strings represent valid numeric values.
2. The numeric value of string `y` is not zero.

(a) Identify the type of error that could occur if these checks are **not** carried out **and** state a cause of this error.

Type ....

....

Cause ....

.... [2]

(b) The designer considers implementing the **checks and calculation** as a module (a procedure or a function). One reason for this is that the same checks and calculations are performed at several places in the program.

Give **another** reason why this is a suitable approach **and** state what is avoided by this approach.

Reason ....

....

Avoided ....

.... [2]

(c) The module to perform the checks and calculation will be implemented as a function. The function will need to return both a real **and** a Boolean value. To achieve this a record type is defined in pseudocode as follows:

```
TYPE Result
   DECLARE Done : BOOLEAN
   DECLARE Value : REAL
ENDTYPE
```

The function `Evaluate()` will:

- take two parameters of type string representing the two numeric values
- return a variable of type `Result` with the `Done` field set to `FALSE` if either of the following applies:
    - at least one of the strings does **not** represent a valid numeric value
    - the numeric value of the string representing value `y` is zero
- otherwise return a variable of type `Result` with the `Done` field set to `TRUE` and the `Value` field assigned the result of the formula (based on the numeric value of the two parameters).

Write pseudocode for the function `Evaluate()`.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]

5 A software developer follows a program development life cycle. The life cycle divides the development process into various stages.

(a) The following table lists some development activities.

Complete the table by writing the name of the life cycle stage for each activity.

Activity	**Name of life cycle stage**
The walkthrough method is used.	
An algorithm is implemented in a programming language.	
The client is interviewed about problems with the current system.	
The program is modified to run on new hardware.	
Records and file structures are defined.	

[5]

(b) The program contains a validation function.

(i) The function will:

- take an integer value as a parameter
- return `TRUE` if the value is within the range 24 to 37, inclusive
- otherwise return `FALSE`.

Complete the table to define a test plan to thoroughly test the operation of the function.

Type of test data	**Test data value**	**Expected result**
Normal	30	TRUE

[4]

(ii) The function is to be tested on its own. When it is shown to work correctly the function will be combined with other modules and testing will continue.

Identify the type of testing that this represents.

.... [1]

BLANK PAGE

6 A factory produces food items. The items must be used within a certain number of days after their production date. The number of days is known as the shelf life. It is different for each type of item but is always a whole number in the range 1 to 21 (inclusive).

The latest date that an item can be used is called the 'use-by' date.

A program is needed to produce labels which show the 'use-by' date.

Part of the program is a function `GetDate()` which will:

- take two parameters: a production date and a value representing the shelf life
- return the corresponding 'use-by' date.

The program contains a global 1D array `DaysInMonth` of type integer which stores the number of days in each month (index 1 is January):

Index	Value
1	`31`
2	`28`
3	`31`
4	`30`
11	`30`
12	`31`

Note: Leap years are **not** considered

(a) An algorithm uses the array `DaysInMonth` to calculate a 'use-by' date. An alternative design would involve the use of multiple selection statements.

An array-based technique is often used when there is a large number of different values to check and where no pattern exists.

One advantage of using an array-based technique is the speed of execution compared to the use of multiple selection statements.

Give **two other** advantages of using an array for this type of operation rather than a solution based on multiple selection statements.

1 ....

....

2 ....

....

[2]

(b) Complete the pseudocode for the function `GetDate()`.

Date functions from the **insert** should be used in your solution.

```
FUNCTION GetDate(ProductionDate : DATE, ShelfLife : INTEGER) RETURNS DATE

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

.......................................................................................

ENDFUNCTION
```

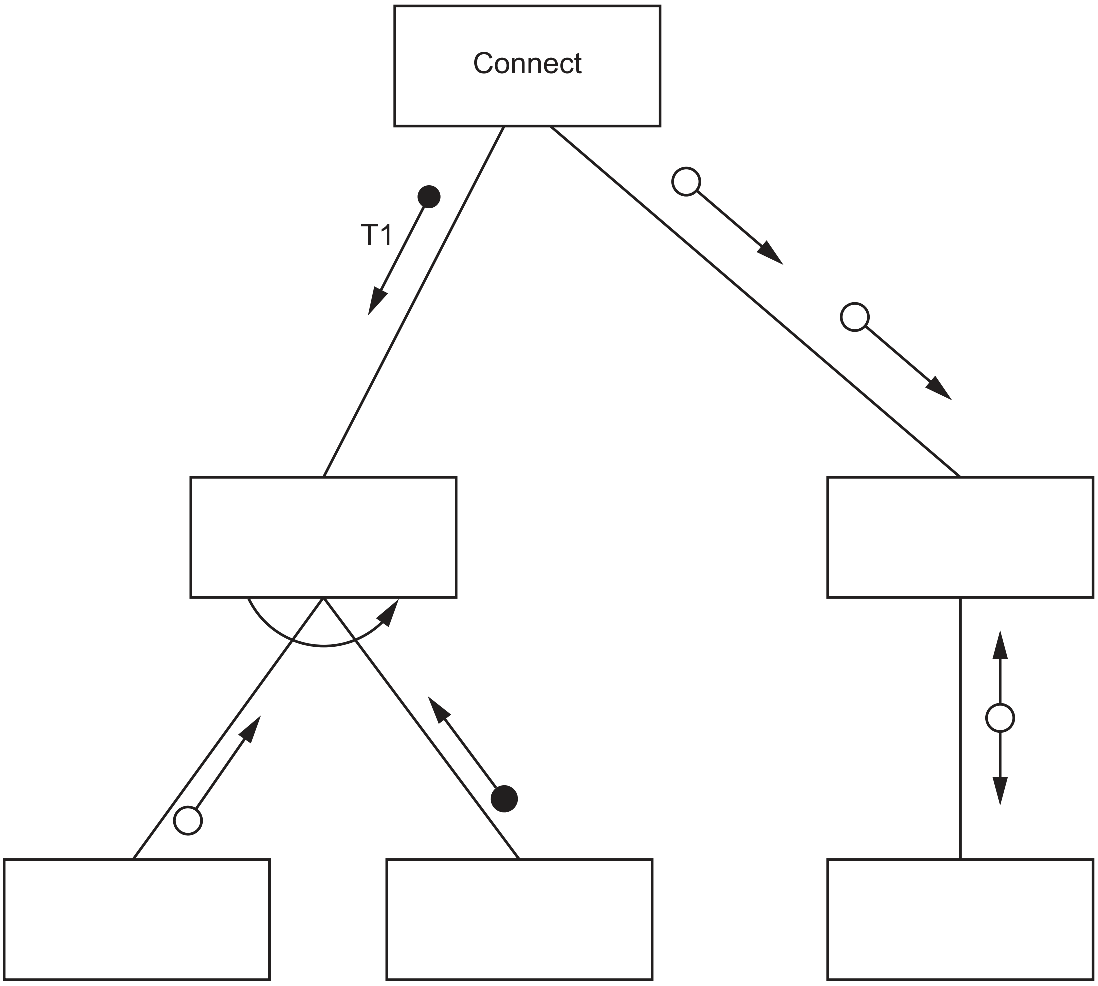
[7]

7 A program contains six modules with headers as follows:

Pseudocode module header
`PROCEDURE Connect()`
`FUNCTION Activate(H1 : STRING, Code : INTEGER) RETURNS BOOLEAN`
`FUNCTION Sync(T1 : BOOLEAN, S2 : REAL) RETURNS INTEGER`
`PROCEDURE Initialise(BYREF ID : INTEGER, BYVAL CC : INTEGER)`
`FUNCTION Reset(RA : STRING) RETURNS INTEGER`
`FUNCTION Enable(SA : INTEGER) RETURNS BOOLEAN`

Module `Connect()` will call either `Activate()` or `Sync()`. This is decided at run-time.

(a) Complete the structure chart for these modules.



[5]

(b) Explain the meaning of the curved arrow symbol used in the diagram in part **(a)**.

....

.... [2]

BLANK PAGE

8 An exam paper has a maximum of 75 marks. One of five pass grades (A to E) is assigned, depending on the mark obtained. The lowest mark for a given grade is known as the grade boundary. For example, if the grade boundary for an A grade is 65 marks, then any candidate who achieves a mark of 65 or above will be awarded an A. A grade of U is awarded for marks below the E grade boundary.

The five grade boundaries are stored in a global 1D array `GradeBoundary` of type integer.

For example:

Element	Value	Comment
`GradeBoundary[1]`	65	The minimum mark for an A grade.
`GradeBoundary[2]`	57	The minimum mark for a B grade.
`GradeBoundary[3]`	43	The minimum mark for a C grade.
`GradeBoundary[4]`	35	The minimum mark for a D grade.
`GradeBoundary[5]`	27	The minimum mark for an E grade.

A global 2D array `Result` of type integer contains candidate marks for the exam. Each row relates to one candidate. Column 1 contains the candidate mark and column 2 contains the unique candidate ID.

For example, for the fourth and fifth candidates:

Element	Mark
`Result[4, 1]`	`56`
`Result[5, 1]`	`54`

Element	ID
`Result[4, 2]`	`1074832`
`Result[5, 2]`	`2573839`

There are more rows in the array than candidates who sit the exam. Any unused rows will be at the end of the array.

Candidate papers that are given a mark within two marks of any grade boundary must be checked.

For example, given the values in the example grade boundaries above, any paper that was awarded between 41 and 45 marks (inclusive) would need to be checked.

A program is being written to identify papers that need to be checked.

The programmer has defined the first program module as follows:

Module	Description
`CheckMark()`	• called with a parameter of type integer representing a candidate mark • returns `TRUE` if the mark is within 2 of any of the five grade boundaries, otherwise returns `FALSE`

(a) Write pseudocode for module `CheckMark()`.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]

(b) A second module is defined:

Module	Description
CheckAll()	• called with a parameter of type integer representing the number of candidate marks in the Result array • uses CheckMark() to check each candidate mark • for each paper that needs to be checked, write the corresponding candidate ID on a separate line in a new file named GRList.txt • outputs a message with a count of how many papers need to be checked

Write pseudocode for module CheckAll().

CheckMark() must be used to check each individual mark.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [8]

(c) The requirement changes. Instead of a new file, the module described in part **(b)** needs to add the corresponding candidate ID for each paper that needs to be checked to an **existing** file.

Explain the change that will need to be made to `CheckAll()`.

....

.... [1]

BLANK PAGE