* 0000800000001 *

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

2213512075

COMPUTER SCIENCE **9618/23**

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.

© UCLES 2024

[Turn over

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

1 (a) The following table contains pseudocode examples.

Each example may contain statements that relate to one or more of the following:

- selection
- iteration (repetition)
- subroutine (procedure or function).

Complete the table by placing **one or more** ticks ('✓') in each row.

Pseudocode example	Selection	Iteration	Subroutine

`FOR Index ← 1 TO 3`
`  IF Safe[Index] = TRUE THEN`
`  Flap[Index] ← 0`
`  ENDIF`
`NEXT Index`			

`CASE OF Compound(3)`			
`REPEAT UNTIL AllDone() = TRUE`			
`WHILE Result[3] <> FALSE`			

[4]

(b) Complete the table by giving the appropriate data type in each case.

Variable	Example data value	Data type
`Available`	`TRUE`	
`Received`	`"18/04/2021"`	
`Index`	`100`	

[3]

(c) Evaluate each expression in the table by using the data values shown in part **(b)**.

Write 'ERROR' if the expression contains an error.

Expression	Evaluates to
`Available AND NOT(Index > 100)`	
`Index MOD 30`	
`NUM_TO_STR(Index + "33")`	

[3]

DO NOT WRITE IN THIS MARGIN

2 An algorithm will:

1. prompt and input a sequence of 100 integer values, one at a time
2. sum the positive integers
3. output the result of the sum.

(a) Write pseudocode for the algorithm.

Assume the value zero is neither positive nor negative.

You must declare all variables used in the algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) The algorithm requires the use of basic constructs. One of these is sequence.

Identify **one other** basic construct required by the algorithm **and** describe how it is used.

Construct .....

Use .....

.....

[2]

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

3 The implementation of a linked list uses an integer variable and a 1D array `List` of type `Node`.

Record type `Node` is declared in pseudocode as follows:

```
TYPE Node
   DECLARE Data : STRING
   DECLARE Pointer : INTEGER
ENDTYPE
```

The array `List` is declared in pseudocode as follows:

```
DECLARE List : ARRAY[1:200] OF Node
```

The linked list will operate as follows:

- Integer variable `HeadPointer` will store the array index for the first node in the linked list.
- The `Pointer` field of a node contains the index value of the next array element (the next node) in the linked list.
- The value 0 is used as a null pointer.

(a) (i) State why the value 0 has been selected as the null pointer.

.....

..... [1]

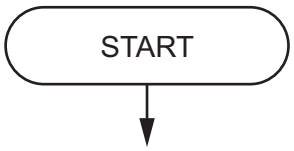
(ii) Give the range of valid values that could be assigned to variable `HeadPointer`.

..... [1]

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) The array `List` will be initialised so that each node points to the following node. The last node will contain a null pointer.

Complete the program flowchart to represent the algorithm for this operation.





[4]

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) An algorithm outputs the `Data` field from all nodes in the array `List`. The order the `Data` is output should be the same order it is stored in the linked list.

Describe the algorithm in **four** steps.

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

Step 1 .....

.....

.....

Step 2 .....

.....

.....

Step 3 .....

.....

.....

Step 4 .....

.....

.....

[4]

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

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

4 An examination 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.

A program is being written to process examination marks.

The five grade boundaries are stored in a global 1D array `GB` of type `INTEGER`, for example:

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

Any paper that achieves a mark within 2 marks of a grade boundary must be checked. Using the given table, a paper with 45 marks would need to be checked.

(a) The pseudocode algorithm to determine whether a paper should be checked is as shown. The mark for the paper is stored in variable `Mark`. Global variables `Mark`, `Index`, `Upper` and `Lower` are declared as integers.

Complete the pseudocode.

```
FOR Index ← 1 TO .....
   Lower ← GB[Index] - 2
   Upper ← .....
   IF Mark ..... AND Mark ..... THEN
      OUTPUT "Check this paper"
   ENDIF
NEXT Index
```

[4]

(b) An alternative algorithm to determine if a paper needs to be checked uses a global 1D array `Check`, containing 76 elements of type `BOOLEAN`. The indices of the array are from 0 to 75 (inclusive), corresponding to the range of possible marks.

An element value in `Check` is `TRUE` if the index is within 2 marks of a grade boundary. For example, in the case where the C grade boundary is 43 the corresponding part of the `Check` array would be as follows:

Index	Value	
40	FALSE	
41	TRUE	
42	TRUE	
43	TRUE	← The grade boundary for a C grade
44	TRUE	
45	TRUE	
46	FALSE	

DO NOT WRITE IN THIS MARGIN

(i) The mark for a given paper is stored in variable `Mark`.

Describe how an algorithm would use the `Check` array to determine whether this paper should be checked.

.....

.....

.....

..... [2]

(ii) A procedure `GBInitialise()` will initialise the `Check` array using values from the `GB` array.

Note it can be assumed that the maximum grade boundary value for A is 70 and the minimum value for E is 15.

Write pseudocode for the procedure.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

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

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
A compiler is used.	
A program that has been released for general use is modified.	
The dry run method is used.	
The program structure is specified.	

[4]

(b) A software developer has written modules `Test_A()` and `Test_B()`. These have been written but contain errors. These modules are called from several places in the main program and testing of the main program (integration testing) has to stop.

Identify a method that can be used to continue testing the main program **before** the errors in these modules have been corrected **and** describe how this would work.

Method .....

Description .....

.....

.....

.....

[3]

DO NOT WRITE IN THIS MARGIN

6 In some countries, on the third Sunday in March, daylight saving time begins when clocks move forward by one hour.

A module `AdjustClock()` will take an integer parameter representing a year. The module will return an integer value representing the number of the day in March on which the clocks move forward.

For example, the following line of pseudocode would assign `DayNumber` the value 20:

```
DayNumber ← AdjustClock(2022)
```

Write pseudocode for the function `AdjustClock()`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [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

7 A coffee shop owner wants to introduce a computerised loyalty card system.

A programmer discusses the details of the system with the shop owner.

(a) Identify the stage of the program development life cycle that this discussion is part of **and** give a document that will be produced during this stage.

Stage .....

Document .....

.....

[2]

(b) The shop will give each customer a loyalty card that displays a unique customer ID as a bar code. A customer will be able to present their card each time they make a purchase. The system will scan the bar code, calculate points, and add them to the customer's total. When the customer next makes a purchase and presents their card, they will have the option to exchange points for a discount.

The designer decides that this activity will be handled by a new module. Decomposition will be used to break the problem of designing the new module down into sub-problems (sub-modules).

Identify **four** sub-modules that could be used in the design of the new module **and** describe their use.

Sub-module 1 .....

Use .....

.....

Sub-module 2 .....

Use .....

.....

Sub-module 3 .....

Use .....

.....

Sub-module 4 .....

Use .....

.....

[4]

DO NOT WRITE IN THIS MARGIN

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

8 A program is being developed to implement a game for up to six players.

During the game, each player assembles a team of characters. At the start of the game there are 45 characters available.

Each character has four attributes, as follows:

Attribute	Examples	Comment
Player	0, 1, 3	The player the character is assigned to.
Role	Builder, Teacher, Doctor	The job that the character will perform in the game.
Name	Bill, Lee, Farah, Mo	The name of the character. Several characters may perform the same role, but they will each have a unique name.
Skill level	14, 23, 76	An integer in the range 0 to 100, inclusive.

The programmer has defined a record type to define each character. The record type definition is shown in pseudocode as follows:

```
TYPE CharacterType
   DECLARE Player : INTEGER
   DECLARE Role : STRING
   DECLARE Name : STRING
   DECLARE SkillLevel : INTEGER
ENDTYPE
```

The `Player` field indicates the player to which the character is assigned (1 to 6). This field value is 0 if the character is **not** assigned to any player.

The programmer has defined a global array to store the character data, as follows:

```
DECLARE Character : ARRAY[1:45] OF CharacterType
```

At the start of the game all record fields are initialised, and all `Player` fields are set to 0

The programmer has defined a program module as follows:

Module	Description

`Count()`	• called with two parameters:
	  ○ an integer representing a player
	  ○ a string representing a character role
	• searches the `Character` array for characters with the given role that are assigned to the given player
	• counts the number of assigned characters and sums their total skill level
	• outputs the result of the search if characters with the given role are found, for example:
	`"Player 3 has 4 characters with the role of Teacher and the total skill level is 65"`
	• if no characters with the given role are found, outputs:
	`"No characters with that role are assigned to this player"`

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 pseudocode for module `Count()`.

```
PROCEDURE Count(ThisPlayer : INTEGER, ThisRole : STRING)
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
ENDPROCEDURE
```

[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

(b) The `Character` array data has been saved in the text file `SaveFile.txt`
Each line of the file contains one element of the array (one record).

New modules are defined:

Module	Description

`Extract()` (already written)	• called with two parameters:
	  ○ a string representing a complete line from the text file
	  ○ an integer representing a field number (see structure below)
	• returns a string representing the required field

`Restore()`	• opens the text file `SaveFile.txt` • reads lines from the file and assigns values to each record in the `Character` array using data from each line of the file

As a reminder, the record structure is repeated here:

```
TYPE CharacterType
   DECLARE Player : INTEGER       //Field number 1
   DECLARE Role : STRING          //Field number 2
   DECLARE Name : STRING          //Field number 3
   DECLARE SkillLevel : INTEGER   //Field number 4
ENDTYPE
```

Write pseudocode for module `Restore()`.

You must use the module `Extract()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

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

.....

.....

.....

.....

..... [7]

(c) The game can last for several days and users often find that they have to close and rerun the game program many times in order to complete it.

Describe the benefit of using the file `SaveFile.txt` as described in part **(b)**.

.....

.....

.....

..... [2]

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

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

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

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

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.