GradeMax

* 0000800000001 *

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

* 0 9 9 5 8 4 1 9 3 4 *

COMPUTER SCIENCE **9618/22**

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 **24** pages. Any blank pages are indicated.

DC (LK) 337008/4
© UCLES 2024

[Turn over

GradeMax

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

1 A program has been developed and released for general use. After a few months of use an error is detected where under certain circumstances it outputs an unexpected value.

(a) The error in the program needs to be corrected.

Identify the stage of the program development life cycle that this correction is made in.

..... [1]

(b) The program contains a function `Lookup()`. After investigation, it is found that this is the function that sometimes returns an incorrect value.

An Integrated Development Environment (IDE) is used to help locate the error.

The IDE features of watch window, single stepping and breakpoint will be used.

Explain these features including the order that they will be used in to locate the error in `Lookup()`.

.....

.....

.....

.....

.....

..... [3]

(c) To solve the error a programmer decides to create a new module.

The design of the new module has been completed and the module is being coded.

Identify **two** features of an IDE that will help during the coding of this new module.

1 .....

2 .....

[2]

(d) The new module referred to in part **(c)** introduces **three** new variables.

Complete the following table by giving the appropriate data type for each.

Variable name	Used to store	Data type
`Name`	A customer name.	
`Index`	An array index.	
`Result`	The result of the division of any two non-zero numbers.	

[3]

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

GradeMax

* 0000800000003 *

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

GradeMax

2 A program is being developed to process bank card information.

When a card number is displayed, all the characters **except** the last four are replaced with the asterisk character `'*'`.

Card numbers are stored as strings. The strings are between 10 and 20 characters in length.

The function `Conceal()` will take a string representing a card number and return a modified string.

Example strings:

Original string	Modified string
`"1234567890"`	`"******7890"`
`"1234567897652"`	`"*********7652"`
`"1234567890123456"`	`"************3456"`

(a) The function `Conceal()` will:

- take a numeric string as a parameter representing the card number
- return a string in which the asterisk character replaces all except the last four characters of the card number parameter.

Write pseudocode for the function `Conceal()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [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

GradeMax

(b) The requirements have been changed. `Conceal()` will now be written as a procedure which will process 100 card numbers each time it is called.

The card numbers will be stored in a 2D array `CardNumber`. The original string will be stored in column one and the modified string in column two.

(i) Write pseudocode to declare the array.

.....

..... [2]

(ii) The new procedure `Conceal()` will write the modified string to the corresponding element in column two.

The array `CardNumber` is passed as a parameter to the new procedure `Conceal()`.

Identify how this parameter should be specified in the new procedure header.

.....

..... [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

GradeMax

3 A program uses a stack to hold up to 60 numeric values.
The stack is implemented using two integer variables and a 1D array.

The array is declared in pseudocode as shown:

```
DECLARE ThisStack : ARRAY[1:60] OF REAL
```

The stack operates as follows:

- Global variable `SP` acts as a stack pointer that points to the next available stack location. The value of `SP` represents an array index.
- Global variable `OnStack` represents the number of values currently on the stack.
- The stack grows upwards from array element index 1.

(a) (i) Give the initial values that should be assigned to the **two** variables.

`SP` .....

`OnStack` .....
[1]

(ii) Explain why it is **not** necessary to initialise the array elements before the stack is used.

.....

.....

.....

..... [2]

(b) A function to add a value to `ThisStack` is expressed in pseudocode as shown.
The function will return a value to indicate whether the operation was successful or not.

Complete the pseudocode by filling in the gaps.

```
FUNCTION Push(ThisValue : REAL) RETURNS BOOLEAN
   DECLARE ReturnValue : BOOLEAN
   IF ..... THEN
      RETURN ..... // stack is already full
   ENDIF
   ..... ← ThisValue
   SP ← .....
   OnStack ← OnStack + 1
   RETURN TRUE
ENDFUNCTION
```

[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

GradeMax

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

GradeMax

4 A global integer variable `Tick` is always incremented every millisecond (1000 times per second) regardless of the other programs running.

The value of `Tick` can be read by any program but the value should not be changed.

Assume that the value of `Tick` does not overflow.

As an example, the following pseudocode algorithm would output `"Goodbye"` 40 seconds after outputting `"Hello"`.

```
DECLARE Start : INTEGER

OUTPUT "Hello"
Start ← Tick

REPEAT
   //do nothing
UNTIL Tick = Start + 40000

OUTPUT "Goodbye"
```

A program is needed to help a user to time an event such as boiling an egg.

The time taken for the event is known as the elapsed time.

The program contains a procedure `Timer()` which will:

- take two integer values representing an elapsed time in minutes and seconds
- use the value of variable `Tick` to calculate the elapsed time
- output a warning message 30 seconds before the elapsed time is up
- output a final message when the total time has elapsed.

For example, to set an alarm for 5 minutes and 45 seconds the program makes the following call:

```
CALL Timer(5, 45)
```

When 5 minutes and 15 seconds have elapsed, the program will output:

```
"30 seconds to go"
```

When 5 minutes and 45 seconds have elapsed, the program will output:

```
"The time is up!"
```

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

GradeMax

Write pseudocode for the procedure `Timer()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [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

GradeMax

5 A program contains a global 1D array `Data` with 100 elements of type `INTEGER`.

The program contains a function `Process()` expressed in pseudocode as follows:

```
FUNCTION Process(Number : INTEGER, Label : STRING) RETURNS STRING
   DECLARE Index, Count : INTEGER
   DECLARE ReturnValue : STRING

   Count ← INT(100 / Number)
   Index ← Data[Number]

   CASE OF (Index MOD 2)
      0 : ReturnValue ← TO_UPPER(RIGHT(Label, Count))
      1 : ReturnValue ← "****"
   ENDCASE

   RETURN RetVal
ENDFUNCTION
```

(a) Run-time errors can be generated in different ways. For example, a run-time error will be generated if a function is called with invalid parameters.

The pseudocode contains three statements that could generate a run-time error.

Write the **three** statements **and** explain how each could generate a run-time error.

Statement 1 .....

Explanation .....

.....

Statement 2 .....

Explanation .....

.....

Statement 3 .....

Explanation .....

.....

[3]

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

* 0000800000011 *

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) One type of run-time error can cause a program to stop responding ('freezing').

Identify a particular type of programming construct that can generate this type of error **and** explain why it occurs.

Construct .....

Explanation .....

.....

.....

[2]

(c) The function `Process()` contains a selection construct using a `CASE` structure.

Write pseudocode using a single selection construct with the same functionality **without** using a `CASE` structure.

.....

.....

.....

.....

..... [2]

GradeMax

6 A shop sells sandwiches and snacks. The owner chooses a 'daily special' sandwich which is displayed on a board outside the shop. Each 'daily special' has two different fillings and is made with one type of bread.

The owner wants a program to randomly choose the 'daily special' sandwich.

The program designer decides to store the possible sandwich fillings in a 1D array of type string.

The array is declared in pseudocode as follows:

```
DECLARE Filling : ARRAY [1:35] OF STRING
```

Each element contains the name of one filling.

An example of the first five elements is as follows:

Index	Element value
1	"Cheese"
2	"Onion"
3	"Salmon"
4	"Anchovies"
5	"Peanut Butter"

A second 1D array stores the possible bread used:

```
DECLARE Bread : ARRAY [1:10] OF STRING
```

Each element contains the name of one type of bread.

An example of the first three elements is as follows:

Index	Element value
1	"White"
2	"Brown"
3	"Pitta"

Both arrays may contain unused elements. The value of these will be an empty string and they may occur anywhere in each array.

A procedure `Special()` will output a message giving the 'daily special' sandwich made from **two** randomly selected different fillings and **one** randomly selected bread.

Unused array elements must **not** be used when creating the 'daily special' sandwich.

Using the above examples, the output could be:

```
"The daily special is Cheese and Onion on Brown bread."
```

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

GradeMax

(a) Complete the pseudocode for the procedure `Special()`.

Assume that both arrays are global.

```
PROCEDURE Special()
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
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

GradeMax

(b) The owner decides that some combinations of fillings do not go well together. For example, anchovies and peanut butter.

Describe how the design could be changed to prevent certain combinations being selected.

.....

.....

.....

.....

.....

.....

.....

..... [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

GradeMax

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 runs a computerised loyalty card system.

Customers are issued with a loyalty card with their name together with a unique customer ID.

Loyalty points are added to their card each time they spend money at the shop.

The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.

A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

(a) Identify **three** items of customer information that will be used by the new module **and** justify your choices.

Item 1 .....

Justification .....

.....

Item 2 .....

Justification .....

.....

Item 3 .....

Justification .....

.....

[3]

(b) Identify the computational thinking skill that you needed to use to answer part **(a)**.

..... [1]

(c) It is decided to adopt a formal program development life cycle model for the development of the new module.

The analysis of the new module is complete and the project moves on to the design stage. During this stage all the necessary algorithms and module designs will be defined.

State **three other** items that will be defined for the new module during the design stage.

1 .....

2 .....

3 .....

[3]

GradeMax

(d) Part of the coffee shop program contains three program modules as follows:

- Module `Init()` has no parameters and returns a Boolean.
- Module `Reset()` takes a string as a parameter and returns an Integer.
- Module `Check()` repeatedly calls `Init()` followed by `Reset()`.

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

[3]

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

GradeMax

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

GradeMax

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.
Level	14, 23, 76	The skill level of the character. 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 Level : INTEGER
ENDTYPE
```

The `Player` field indicates the player to which the character is assigned (1 to 6). The 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` field values are set to 0

The programmer has defined a program module as follows:

Module	Description

`Assign()`	• called with two parameters:
	  ○ an integer representing a player
	  ○ a string representing a character role
	• search the `Character` array for an unassigned character with the required role
	• If found, assign the character to the given player and output a confirmation message, for example:
	  `"Bill the Builder has been assigned to player 3"`
	• If no unassigned character with the required role is found, output a suitable message.

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

GradeMax

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

.....

.....

.....

GradeMax

(b) A new module will store the contents of the `Character` array in a text file.

The module is defined as follows:

Module	Description
`Save()`	• form a string from each record with fields separated by the character `'^'` • write each string to a separate line of the new file named `SaveFile.txt`

Complete the pseudocode for module `Save()`.

```
PROCEDURE Save()
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
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

(c) The program is changed and the record type definition is modified as follows:

```
TYPE CharacterType
   DECLARE Player : INTEGER
   DECLARE Role : STRING
   DECLARE Name : STRING
   DECLARE Level : INTEGER
   DECLARE Status : BOOLEAN
ENDTYPE
```

Describe how the additional Boolean field may be stored with the rest of the fields on one line of a text file.

.....

.....

.....

..... [2]

(d) The save operation is to be extended so that multiple files may be saved as the game progresses. This will allow the user to restore the game from any saved position. The filename must reflect the sequence in which the files are saved.

Describe a method that would allow multiple files to be saved **and** give an example of **two** consecutive filenames.

Method .....

.....

Example .....

.....

[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

GradeMax

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

GradeMax

* 0000800000023 *

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

© UCLES 2024

GradeMax

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.