GradeMax

* 0000800000001 *

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

4804981983

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 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 **20** pages. Any blank pages are indicated.

DC (PQ/CT) 342887/5
© UCLES 2025

GradeMax

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

1 A program is being developed for the management of a sports centre.

(a) The programmer developing the software decides to use modules (procedures or functions) and local variables. They also decide not to use global variables.

(i) State **two** reasons why the programmer decides to use modules.

1 .....

.....

2 .....

.....

[2]

(ii) State **one** difference between local and global variables.

.....

..... [1]

(iii) State **two** benefits of using local variables.

1 .....

.....

2 .....

.....

[2]

(b) The pseudocode design contains a number of expressions.

Complete each pseudocode expression so that it evaluates to the value shown.

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

Expression	Evaluates to
.....(68)	'D'
.....(04/02/2025)	2
.....TRUE	FALSE
.....(..... ("Court1.4Upper" ,.....,.....))	1.4

[5]

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 table lists some of the variables used in the program.

Complete the table by writing the most appropriate data type for each variable.

Variable	Use of variable	Data type
`MemberCount`	stores how many members have used the sports centre each day	
`TotalTakings`	stores the total amount of money taken each day	
`BookingConfirmed`	stores the state of a booking for an exercise class; a booking is either confirmed or **not** confirmed	
`MemberDOB`	to calculate when to send an email with a birthday message to a member	

[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

2 A programmer uses a number of Abstract Data Types (ADT) in the programs that she is developing.

(a) A stack using memory locations 400 to 409 is used in one program.

The diagram represents the current state of the stack.

A variable `TopOfStack` pointer indicates the last value added to the stack.

Stack

Memory location	Value	
409		
408		
407		
406		
405		
404		
403	'P'	← `TopOfStack`
402	'X'	
401	'D'	
400	'B'	

(i) Complete the answer column in the following table:

	Answer
the memory location of the value that has been on the stack for the longest time	
the number of consecutive **push** operations that will result in the `TopOfStack` variable containing 409	

[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

(ii) The diagram shows the current state of the stack.

Stack

Memory location	Value	
409		
408		
407	'A'	← TopOfStack
406	'C'	
405	'F'	
404	'K'	
403	'B'	
402	'S'	
401	'R'	
400	'D'	

The following sequence of operations are performed:

```
PUSH 'T'
POP
POP
PUSH 'Z'
PUSH 'X'
POP
PUSH 'Y'
```

Complete the following diagram to show the state of the stack after the operations have been performed.

Stack

Memory location	Value
409	
408	
407	
406	
405	
404	
403	
402	
401	
400	

[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

(b) In a different program, the programmer decides to implement a linked list using an array of records.

(i) The array is represented in the diagram.

A pointer value of `-1` indicates the end of the linked list.

Complete the pointer field to produce a linked list that is in alphabetical order.

Index	Data	Pointer
0	"Neptune"	
1	"Jupiter"	
2	"Saturn"	
3	"Earth"	
4	"Mercury"	
5	"Uranus"	

[3]

(ii) The variable `StartPointer` contains the index of the first item in the linked list.

State the value of the variable `StartPointer`

..... [1]

(c) A third program is also being created to manage a queue.

Describe **two** features of a queue.

Feature one .....

.....

Feature two .....

.....

[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

3 A programmer has been asked to create a module `RollDice()` to simulate multiple rolls of a dice. This module will be used as part of a program for a game.

The module will:

step 1 – take a positive integer parameter representing the number of times the dice will be rolled

step 2 – simulate **one** roll of the dice by generating a random integer between 1 and 6 inclusive

step 3 – output the integer generated

step 4 – repeat, as required from **step 2**

step 5 – calculate the average value of the random integers generated

step 6 – return the average value.

Write pseudocode for the module `RollDice()`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

GradeMax

4 A programmer is developing a new stock control program for a shop owner to produce sales reports.

(a) A text file `Sales.txt` stores strings representing the number of items sold.

The file contains 52 lines, each representing the number of items sold in each week of the year.

For example:

File line number	File data
1	"350"
2	"502"
3	"434"
49	"492"
50	"872"
51	"772"
52	"201"

The example shows that 502 items were sold in week 2.

The owner requires a module to output all week numbers where the number of items sold is greater than 500.

Describe an algorithm that produces the required module.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

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) Once the program has been completed, it is tested using the walkthrough method.

Describe the walkthrough method and explain how it can be used to identify errors.

.....

.....

.....

.....

.....

.....

.....

..... [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

5 A module `Parity()` takes a string as a parameter. The parameter has the identifier `BitString` and it represents a binary value.

The module will concatenate a single character to the end of `BitString` by applying one of the two rules:

- `'0'` if `Bitstring` contains an even number of 1s
- `'1'` if `Bitstring` contains an odd number of 1s.

The modified value of `BitString` is then returned.

For example:

Parameter	String returned	Explanation
`"0010010110"`	`"00100101100"`	there are an even number of 1s in the string passed to `Parity()` so a `'0'` is concatenated to the end of `BitString`
`"101010"`	`"101010`**`1`**`"`	there are an odd number of 1s in the string passed to `Parity()` so a `'1'` is concatenated to the end of `BitString`

Write pseudocode for the module `Parity()`

Assume the parameter `BitString` can only contain the characters `'0'` and `'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

.....

.....

.....

.....

.....

.....

.....

.....

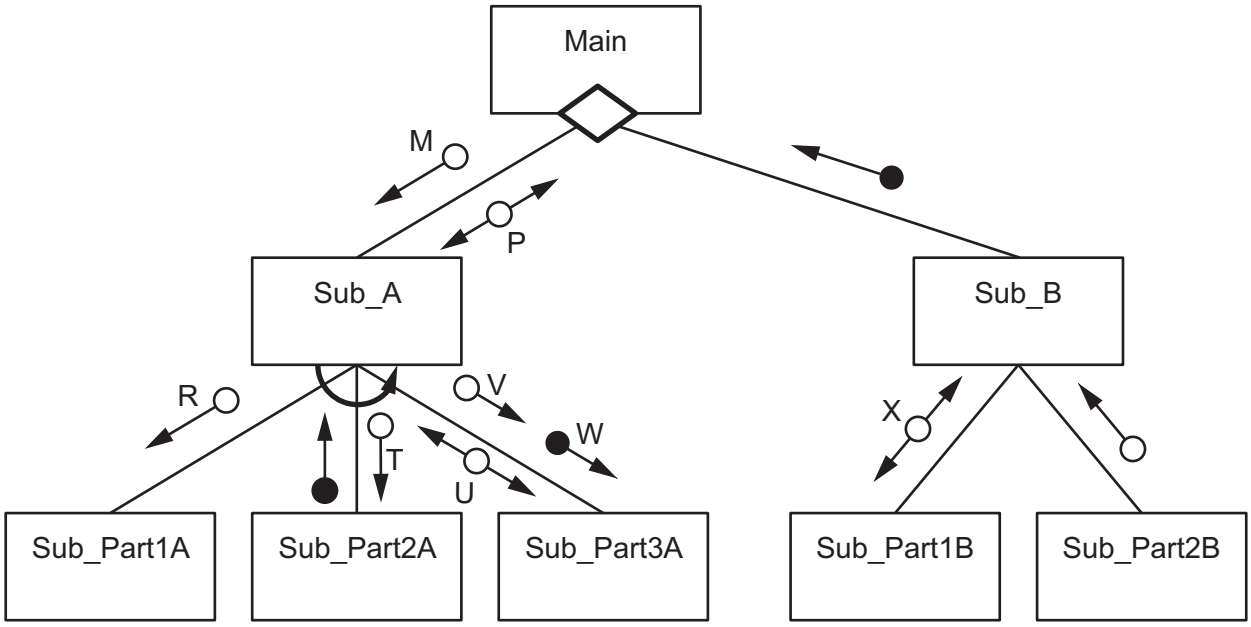
.....

..... [8]

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

6 Study the structure chart:



Some of the parameter data types are:

- R and V are of type `INTEGER`
- T is of type `REAL`
- U is of type `STRING`
- W is of type `BOOLEAN`.

Some of the modules in the structure chart are functions, others are procedures.

(a) Explain **one** difference between functions and procedures.

.....

..... [1]

(b) Write the pseudocode to define the module headers:

Sub_Part1A

.....

.....

Sub_Part2A

.....

.....

Sub_Part3A

.....

..... [5]

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

(c) The structure chart uses the two symbols shown.

Complete the table to explain what each symbol means and how the relevant modules in the structure chart are affected.

Symbol	Explanation
	....
	....

[3]



GradeMax

7 A program is being developed to implement a customer loyalty scheme for a coffee shop.

The programmer has decided that the following data items need to be stored for each customer:

Data item	Description
customer ID	a unique six-digit string
loyalty points	an integer value that depends on how often the customer visits the coffee shop
last visit date	the date the customer last visited the coffee shop

The programmer has defined a program module:

Module	Description
`UpdateVisit()`	• called with two parameters: 1. loyalty points of type `INTEGER` 2. last visit date of type `DATE` • change the loyalty points: o increase by **four** if the last visit date and the current date are in the same month o otherwise increase by **one** • change the last visit date to the current date • the changed values must be available to the code that follows the call to `UpdateVisit()`

(a) (i) Write pseudocode for module `UpdateVisit()`

Assume the customer has made at least **one** previous visit.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

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

(ii) The programmer decides to amend the `UpdateVisit()` module so that loyalty points are further increased if the customer visits the coffee shop the same day of the week as their last visit.

Write the pseudocode for this condition.

.....

..... [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

(b) A text file `Loyalty.txt` will be used to store the data items for the loyalty scheme.

The text file `Loyalty.txt` contains only one line of data for each customer.

Each data item is separated by a comma:

`<CustomerID>,<LoyaltyPointsString>,<LastVisitDateString>`

`LastVisitDateString` is always eight characters in length in the format: DDMMYYYY.

An example of how a line of data will be stored in the text file `Loyalty.txt` is:

```
"100123,132,05032025"
```

This example shows that the customer with an ID of 100123 had 132 loyalty points following their last visit to the coffee shop on 05/03/2025.

If a customer visits the coffee shop on a Monday, the value of their loyalty points is increased by 10.

The programmer has defined **two** more program modules:

Module	Description
`FindCustomer()` (is already written)	• called with a parameter of type `STRING` that represents a customer ID • returns the line of data read from the text file `Loyalty.txt` containing the data items for that customer
`MondayCheck()`	• called with a parameter of type `STRING` that represents the customer ID of a customer • calls `FindCustomer()` • extracts the loyalty points from the string returned by `FindCustomer()` • increases the loyalty points for the current customer by 10 if the day visited is a Monday • returns an integer value representing the loyalty points

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

(i) Write pseudocode for module `MondayCheck()`

The module `FindCustomer()` must be used and assume it returns a valid string for the current customer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

GradeMax

(ii) A date is stored in `LastVisitDateString` in the format:

DDMMYYYY

DD is a 2-digit string
MM is a 2-digit string
YYYY is a 4-digit string.

For example, the date 03/09/2024 is stored as `"03092024"`

An algorithm is needed to convert the string stored in `LastVisitDateString` to data type `DATE` which is then stored in the variable `LastVisitDate`.

Complete the pseudocode for this algorithm:

Assume that `LastVisitDateString` has been declared and contains valid data.

```
DECLARE DayInt, MonthInt, YearInt : INTEGER

DECLARE LastVisitDate : DATE
```

.....

.....

.....

.....

.....

.....

.....

.....

[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

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.