Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER | | | | | |

CANDIDATE NUMBER | | | | |

5334515327

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2022**

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 is required for a shopping website.

(a) Part of the program requires four variables. The following table describes the use of each variable.

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

Variable use	Data type
Store the number of days in the current month	INTEGER
Store the first letter of the customer's first name	CHAR
Store an indication of whether a year is a leap year	BOOLEAN
Store the average amount spent per customer visit	REAL

[4]

(b) The designer considers the use of a development life cycle to split the development of the website into several stages.

(i) State **one** benefit of a development life cycle when developing the website.

Easier to manage/plan/cost // Clear deliverables produced at (end of) each stage

[1]

(ii) Analysis is one stage of a development life cycle.

State **one** document that may be produced from the analysis stage of the website project.

- The problem definition
- Requirements specification // Client requirements
- Documentation related to current system (e.g. ER diagram of current system, DFD of current system, feasibility study)

[1]

(c) The program will be developed using the Rapid Application Development (RAD) life cycle.

(i) State **one** principle of this life cycle.

- Modules are developed in parallel / as prototypes
- Minimal / no detailed planning is carried out // Allows for changes to requirements
- Flexible development process
- Small incremental releases are made, each adding functionality
- Used for time critical development
- Client involved during (all stages) of development

..... [1]

(ii) Give **two** benefits and **one** drawback of its use compared to the waterfall life cycle.

Benefit 1 .....

Benefit 2 .....

Drawback .....

Benefits: **(Max 2 marks)**
- Quicker development possible / Multiple areas can be worked on at same time
- Prototype produced (at early stage in process)
- Easier to change requirements / quicker delivery of usable modules
- Early review possible / closer cooperation between client and developers

Drawback: **(Max 1 mark)**
- Difficult to estimate cost / time to complete project
- Documentation often omitted
- Lack of client availability throughout life cycle // too easy for client to keep changing their mind

[3]

(d) Adaptive maintenance needs to be carried out on the website program.

Give **two** reasons why adaptive maintenance may be required.

1 .....

2 .....

1 Change to website requirements
2 New technologies available to host website // changes made to library modules used
3 Change in relevant legislation

[2]

2 A program is being designed for a smartphone to allow users to send money to the charity of their choice.

Decomposition will be used to break the problem down into sub-problems.

Identify **three** program modules that could be used in the design **and** describe their use.

Module 1 .....

Use .....

.....

.....

.....

Module 2 .....

Use .....

.....

.....

.....

Module: `SelectCharity()`
Use: Allows the user to choose a particular charity

Module: `SpecifyAmountAndType()`
Use: Allows the user to specify a single or regular payment

Module: `MakePayment()`
Use: Make payment to the charity

Module: `ValidatePayment()`
Use: Validate payment details (by accessing bank computer)

Module: `AddBankAccountDetails() / AddPaymentDetails()`
Use: Allows the user to add bank account information that donation to be taken from

Module: `AddDonorDetails()`
Use: Allows user to add details such as name and contact details

Module 3 .....

Use .....

.....

.....

.....

[3]

3 An algorithm is needed to process a sequence of numbers. Numbers may be positive or negative and may be integer or decimal.

The algorithm will:

- prompt and input one number at a time until the value zero is input
- sum the negative numbers
- sum the positive numbers
- when zero is input, output the two sum values and then end.

Describe the algorithm needed. Do **not** include pseudocode statements in your answer.

1 Declare two (`REAL`) variables for the two sum values **AND** initialise both to zero
2 Prompt **AND** Input a number
3 If number greater than zero add to positive sum and If number less than zero add to negative sum
4 Repeat from step 2 if number not zero
5 After loop the Output `SumPos` and `SumNeg`

..... [5]

4 **(a)** A program contains a 1D array `DataItem` with 100 elements.

State the **one additional** piece of information required before the array can be declared.

The data type (of the item to be stored)

[1]

(b) A programmer decides to implement a queue Abstract Data Type (ADT) in order to store characters received from the keyboard. The queue will need to store at least 10 characters and will be implemented using an array.

(i) Describe **two** operations that are typically required when implementing a queue. State the check that must be carried out before each operation can be completed.

Operation 1

Operation: Add an item / Enqueue

Check 1 Check: There are unused elements in the array // The queue is not full

Operation: Remove an item / Dequeue

Check: There are items in the array // The queue is not empty

Operation 2

Operation: Add an item / Enqueue

Check 2 Check: There are unused elements in the array // The queue is not full

Operation: Remove an item / Dequeue

Check: There are items in the array // The queue is not empty

[4]

(ii) Describe the declaration and initialisation of the variables and data structures used to implement the queue.

1 Declare a (1D) array of size >= 10
2 …of data type `CHAR`
3 Declare integer variable for `FrontOfQueuePointer`
4 Declare integer variable for `EndOfQueuePointer`
5 Initialise `FrontOfQueuePointer` and `EndOfQueuePointer` to represent an empty queue
6 Declare integer variable or `NumberInQueue`
7 Declare integer variable for `SizeOfQueue` to count / limit the max number of items allowed // Reference to mechanism for defining 'wrap' of circular queue
8 Initialise `SizeOfQueue` // Initialise `NumberInQueue`

[5]

5 **(a)** A text string contains three data items concatenated as shown:

<StockID><Description><Cost>

Item lengths are:

Item	Length
StockID	5
Description	32
Cost	the remainder of the string

A procedure `Unpack()` takes four parameters of type string. One parameter is the original text string. The other three parameters are used to represent the three data items shown in the table and are assigned values within the procedure. These values will be used by the calling program after the procedure ends.

(i) Write pseudocode for the procedure `Unpack()`.

```
PROCEDURE UnPack(BYVAL TLine : STRING, BYREF SID, SDesc,
SCost : STRING)
  SID ← LEFT(TLine, 5)
  SDesc ← MID(TLine, 6, 32)
  SCost ← RIGHT(TLine, LENGTH(TLine) - 37)
ENDPROCEDURE
```

[6]

(ii) Explain the term **procedure interface** with reference to procedure `Unpack()`.

1 Provides a mechanism to allow calling program to pass data
2 Defines the four parameters of `Unpack()`
3 … giving their data type and order

[2]

(b) The design changes and a record structure is defined to store the three data items.

A user-defined data type `StockItem` is created as shown:

```
TYPE StockItem
   DECLARE StockID : STRING
   DECLARE Description : STRING
   DECLARE Cost : REAL
ENDTYPE
```

(i) A variable `LineData` of type `StockItem` is declared.

Write the pseudocode statement to assign the value `12.99` to the `Cost` field of `LineData`.

.... `LineData.Cost ← 12.99` .... [1]

(ii) Procedure `Unpack()` is modified and converted to a function which takes the original text string as the only parameter.

Explain the other changes that need to be made to convert the procedure into a function.

- The new function will return an item of type `StockItem`
- Need to declare/use a (local) variable of type `StockItem`
- `Cost`field needs to be converted from a string to a real

.... [2]

(c) `Unpack()` is part of a program made up of several modules. During the design stage, it is important to follow good programming practice. One example of good practice is the use of meaningful identifier names.

Give the reason why this is good practice. Give **two other** examples of good practice.

Reason

Makes the code easier to understand // Describes the purpose of the identifier // Makes the code easier to debug/test/maintain

Example 1

- White space
- Indentation
- Keywords in capitals
- Comments
- Local variables // parameters

Example 2

[3]

(d) The program that includes `Unpack()` is tested using the walkthrough method.

Describe this method **and** explain how it can be used to identify an error.

1 The program is checked by creating a trace table / going through the program a line at a time
2 ….to record/check variable (values) as they change
3 Error may be indicated when variable given an unexpected value
4 Error may be indicated by an unexpected path through the program // Faults in the logic of the program can be detected

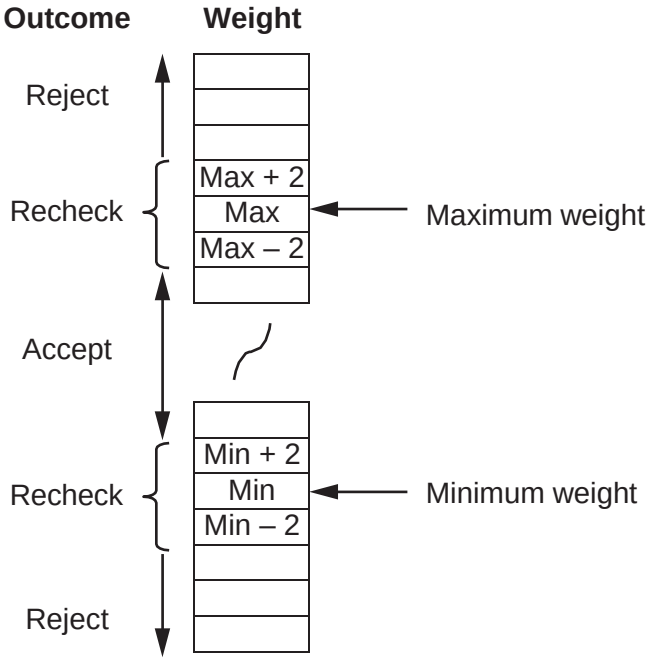
[3]

6 Components are weighed during manufacture. Weights are measured to the nearest whole gram.

Components that weigh at least 3 grams more than the maximum weight, or at least 3 grams less than the minimum weight, are rejected.

A component is rechecked if it weighs within 2 grams of either the maximum or minimum weight.

The final outcome of weighing each component is shown below:



A function `Status()` will be called with three parameters. These are integers representing the weight of an individual component together with the minimum and maximum weights.

The value returned from the function will be as follows:

Outcome	Return value
Accept	`'A'`
Reject	`'R'`
Recheck	`'C'`

(a) Complete the following test plan for **five** tests that could be performed on function `Status()`. The tests should address all possible outcomes.

Test number	Component weight	Min	Max	Expected return value
1	300	290	315	'A'
2	> 317	290	315	'R'
3	317	290	315	'C'
4	288	290	315	'C'
5	< 288	290	315	'R'

[5]

(b) Write pseudocode for `Status()`.

```
Function Status(Actual, Min, Max : INTEGER) RETURNS CHAR
  DECLARE Result : CHAR
  CONSTANT Accept = 'A'
  CONSTANT Reject = 'R'
  CONSTANT ReCheck = 'C'

  Result ← ReCheck

  //Check if reject
  IF Actual > Max + 2 OR Actual < Min - 2 THEN
     Result ← Reject
  ENDIF

  //Check if acceptable
  IF Actual < Max - 2 AND Actual > Min + 2 THEN
     Result ← Accept
  ENDIF


  RETURN Result
ENDFUNCTION
```

[6]

7 A teacher is designing a program to perform simple syntax checks on programs written by students.

Two global 1D arrays are used to store the syntax error data. Both arrays contain 500 elements.

- Array `ErrCode` contains integer values that represent an error number in the range 1 to 800.
- Array `ErrText` contains string values that represent an error description.

The following diagram shows an example of the arrays.

Index	ErrCode	ErrText
1	10	"Invalid identifier name"
2	20	"Bracket mismatch"
3	50	""
4	60	"Type mismatch in assignment"
...		
500	999	<Undefined>

Note:

- There are less than 500 error codes so corresponding elements in both arrays may be unused. Unused elements in `ErrCode` have the value 999. These will occur at the end of the array. The value of unused elements in `ErrText` is undefined.
- Values in the `ErrCode` array are stored in ascending order but not all values may be present. For example, there may be no error code 31.
- Some error numbers are undefined. In these instances, the `ErrCode` array will contain a valid error number but the corresponding `ErrText` element will contain an empty string.

The teacher has defined one program module as follows:

Module	Description

`OutputRange()`	• Prompts for input of two error numbers
	• Outputs a list of error numbers between the two numbers input (inclusive) together with the corresponding error description
	• Outputs a warning message when the error description is missing as for error number 50 in the example
	• Outputs a suitable header and a final count of error numbers found
	
	Output based on the example array data above:
	
	`List of error numbers from 1 to 60`
	
	`10 : Invalid identifier name`
	`20 : Bracket mismatch`
	`50 : Error Text Missing`
	`60 : Type mismatch in assignment`
	
	`4 error numbers output`

(a) Write pseudocode for module `OutputRange()`. Assume that the two numbers input represent a valid error number range.

```
PROCEDURE OutputRange()
  DECLARE First, Last, Count, Index, ThisErr : INTEGER
  DECLARE ThisMess : STRING
  DECLARE PastLast: BOOLEAN

  Count ← 0
  Index ← 1
  PastLast ← FALSE

  OUTPUT "Please input first error number: "
  INPUT First
  OUTPUT "Please input last error number: "
  INPUT Last

  OUTPUT "List of error numbers from ", First, " to ",
Last

  WHILE Index < 501 AND NOT PastLast
     ThisErr ← ErrCode[Index]
     IF ThisErr > Last THEN
        PastLast ← TRUE
     ELSE
        IF ThisErr >= First THEN
           ThisMess ← ErrText[Index]
           IF ThisMess =  "" THEN
              ThisMess ← "Error Text Missing"
           ENDIF
           OUTPUT ThisErr, " : ", ThisMess
           Count ← Count + 1
        ENDIF
     ENDIF
     Index ← Index + 1
   ENDWHILE

  OUTPUT Count, " error numbers output"

ENDPROCEDURE
```

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

Question 7 continues on the next page.

(b) (i) Two additional modules are defined:

Module	Description
`SortArrays()`	• Sorts the arrays into ascending order of `ErrCode`

`AddError()`	• Takes two parameters:
	  ◦ an error number as an integer
	  ◦ an error description as a string
	• Writes the error number and error description to the first unused element of the two arrays. Ensures the `ErrCode` array is still in ascending order
	• Returns the number of unused elements **after** the new error number has been added
	• Returns −1 if the new error number could not be added

Write pseudocode for the module `AddError()`. Assume that the error code is **not** already in the `ErrCode` array.

```
FUNCTION AddError(ErrNum : INTEGER, ErrMess : STRING)
RETURNS INTEGER
  DECLARE Index, Remaining : INTEGER
  CONSTANT Unused = 999

  Index ← 1
  Remaining ← -1

  REPEAT
     IF ErrCode[Index] = Unused THEN
        ErrCode[Index] ← ErrNum
        ErrText[Index] ← ErrMess
        CALL SortArrays()
        Remaining ← 500 - Index
     ENDIF
     Index ← Index + 1
  UNTIL Remaining <> -1 OR Index > 500

  RETURN Remaining

ENDFUNCTION
```

.....

.....

.....

.....

.....

.....

..... [6]

(ii) A new Module `RemoveError()` will remove a given error number from the array.

Describe the algorithm that would be required. Do **not** include pseudocode statements in your answer.

1. Loop through 500 elements (while error number not found)
2. Compare `ErrCode` for current element with the error number
3. If same, set element value to 999 (and terminate loop)
4. ... and call `SortArrays()` (to move 999 to the end) – once only

.....

.....

.....

..... [3]

BLANK PAGE

BLANK PAGE

BLANK PAGE