

Candidate Marks Report

Series : B 2025

This candidate's script has been assessed using On-Screen Marking. The marks are therefore not shown on the script itself, but are summarised in the table below.

Centre No :	PK470	Assessment Code :	9618
Candidate No :	65	Component Code :	22
Candidate Name :	MUHAMMAD UMAR ISLAM,		

In the table below 'Total Mark' records the mark scored by this candidate.

'Max Mark' records the Maximum Mark available for the question.

Paper:	9618/22	
Paper	62 / 75	
Total:		
Question	Total / Max Mark Mark	
1a	4 / 4	
1b	3 / 3	
1c	4 / 4	
2	5 / 5	
3a	6 / 6	
3b	1 / 1	
3c	1 / 3	
4a	3 / 4	
4b	4 / 5	
4c	0 / 4	
5a	4 / 4	
5b	1 / 1	
6	6 / 7	
7a	1 / 2	
7b	4 / 4	
8a	7 / 7	
8b	6 / 7	
8c	0 / 2	
8d	2 / 2	



Cambridge International AS & A Level

CANDIDATE
NAME

MUHAMMAD UMAR ISLAM

CENTRE
NUMBER

P K 4 7 0

CANDIDATE
NUMBER

0 0 6 5

COMPUTER SCIENCE

9618/22

Paper 2 Fundamental Problem-solving and Programming Skills

October/November 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.

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

- 1 (a) The table contains pseudocode examples.

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

- selection
- iteration (repetition)
- subroutines (procedures or functions).

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

Pseudocode example	Selection	Iteration	Subroutine
IF Status = FALSE THEN FOR Count ← 1 TO 20 CALL Reset(Count) NEXT Count ENDIF	✓	✓	✓ ✓
OTHERWISE : Status ← TRUE	✓		✓
WHILE AllDone() = TRUE		✓	✓ ✓
30 : NextChar ← 'X'	✓		✓

✓[4]

- (b) Complete the table by giving the appropriate data type.

Variable	Example data value	Data type
Result	5.42	REAL ✓
MonthLetter	"JFMAMJJASOND"	STRING ✓
Birthday	15/11/2009	DATE ✓

[3]

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

Write 'ERROR' if the expression contains an error.

Expression	Evaluates to
INT(Result) + 1 > 6	FALSE ✓
NUM_TO_STR(LENGTH(MonthLetter))	"12" ✓
NUM_TO_STR(Result + "3.2")	ERROR ✓
MID(MonthLetter, MONTH(Birthday) - 2, 1)	"S" ✓

[4]

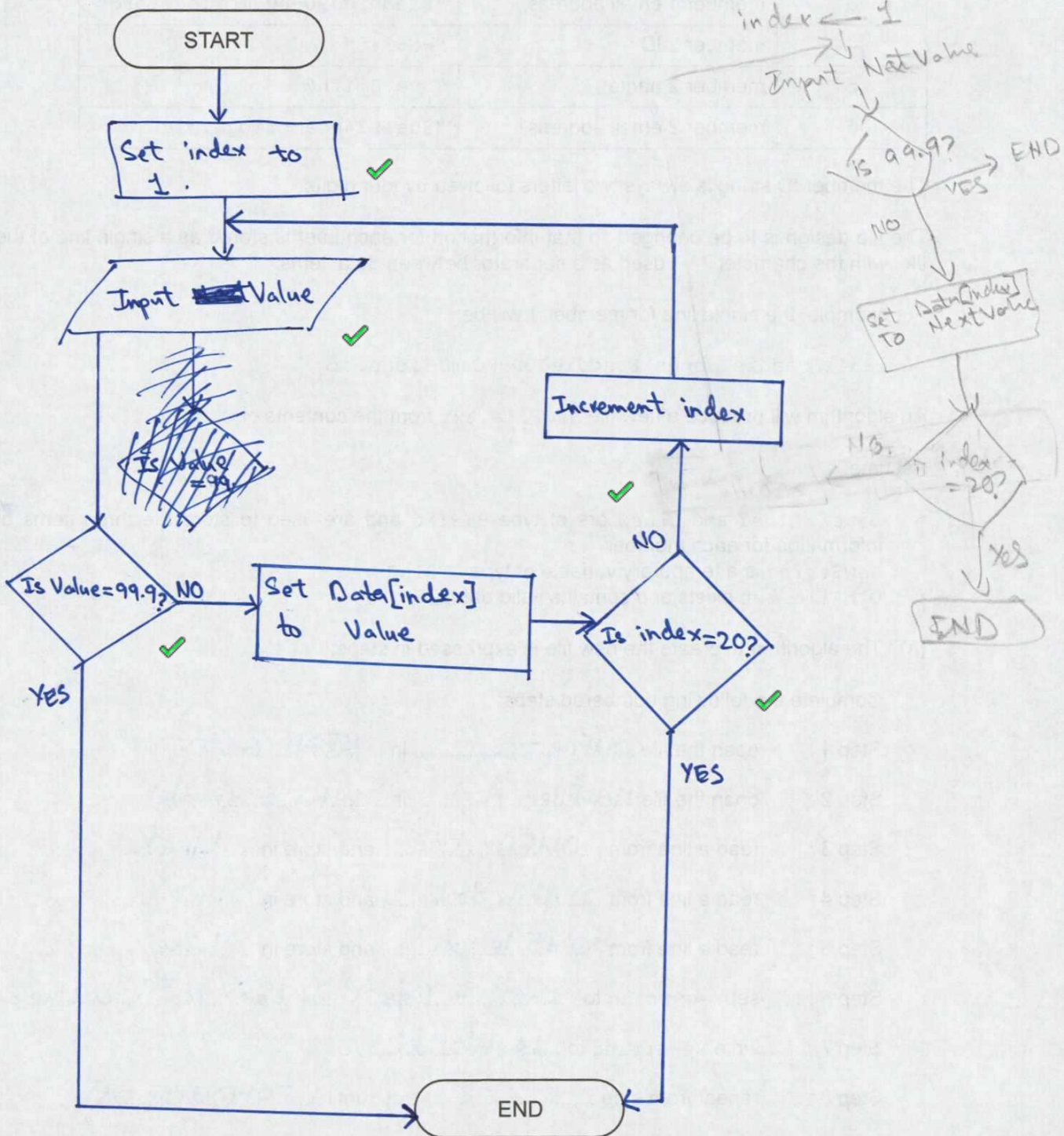


2 Data is a global 1D array containing 20 elements of type REAL

An algorithm will:

- input a sequence of real values, one at a time
- assign each value to consecutive array elements, starting from index 1
- end when the value 99.9 is input, or all 20 elements have been assigned (the value 99.9 must **not** be stored in the array).

Complete the program flowchart to represent the algorithm:





- 3 A text file `OldFile.txt` contains IDs, names and email addresses for members of a club. Three information items are stored for each member and each item is stored on a separate line.

The example shows the information for the first two members in the first six lines of the file:

Line in file	Information item	Example data
1	member 1 ID	"AB1234"
2	member 1 name	"Freddie Jones"
3	member 1 email address	"FreddieJ909@Cambridge.org"
4	member 2 ID	"BC2345"
5	member 2 name	"Sue Smith"
6	member 2 email address	"Sue1024@Cambridge.org"

The member ID string is always two letters followed by four digits.

The file design is to be changed so that information for each user is stored as a single line of the file, with the character '`\`' used as a separator between data items.

For example, the single line for member 1 will be:

"AB1234\Freddie Jones\FreddieJ909@Cambridge.org"

An algorithm will produce a new file `NewFile.txt` from the contents of `OldFile.txt`

Assume:

- `LineX`, `LineY` and `LineZ` are of type `STRING` and are used to store the three items of information for each member
- `NewString` is a temporary variable of type `STRING`
- `OldFile.txt` exists and contains valid data.

- (a) The algorithm to create the new file is expressed in steps.

Complete the following numbered steps:

- Step 1: open the file `OldFile.txt` in `READ (mode)` ✓
- Step 2: open the file `NewFile.txt` in `WRITE (mode)` ✓
- Step 3: read a line from `OldFile.txt` and store in `LineX`
- Step 4: read a line from `OldFile.txt` and store in `LineY`
- Step 5: read a line from `OldFile.txt` and store in `LineZ` ✓
- Step 6: set `NewString` to `LineX & '\ ' & LineY & '\ ' & LineZ`
- Step 7: write `NewString` to `NewFile.txt` ✓
- Step 8: repeat from step `3` until `EOF(OldFile.txt)` ✓
- Step 9: close both files.





- (b) The character '\ ' has been chosen as a separator.

Explain why this is a suitable character.

this character is not used anywhere in data, not in member's ID, name or address, so is best to indicate the separation. [1]

- (c) Two new information items are to be stored for each member. Both new items are encrypted; they can each contain any character (including '\ ') and can be of any length up to a maximum of 99 characters.

For example:

Information item	Example data
member 1 ID	"AB1234"
member 1 name	"Freddie Jones"
member 1 email address	"FreddieJ909@Cambridge.org"
member 1 new information 1	"En/98&* (? \ / D7iP"
member 1 new information 2	"23 \ CoboL"

Explain the additional file design changes needed so that:

- all the information items for each member are stored on a single line of NewFile.txt
- it is possible to extract each information item after the line is read.

Assume the '\ ' character is not used in any email address.

When adding new information data to the line, keep the older information items as they were, (use '\ ' to separate them and add a ~~check~~ counter to check the two characters (1) ~~data~~ found) the first two items will be extracted easily and before adding information data, ~~add~~ concatenate a string(newinfo) before each one and a length check, so when this string(newinfo) is found, the data before that string and after second '\ ' is third information item and after first string(newinfo) before second string(newinfo), the data in between is member's first new information, likewise after second string(newinfo) found, the remaining data is member's second new information. [3]





4 A quiz has nine questions. There are:

- five easy questions each worth 3 points $5 \times 3 = 15$
- four hard questions each worth 5 points. $5 \times 4 = 20$

At the end of the quiz the points for each correctly answered question are added to give a total score.

A check is made to test that the total score is valid using a 1D array `CheckTotal` of type `BOOLEAN`. Each index value of the array represents a possible total score. The corresponding element value is `TRUE` if the index value represents a valid total score and `FALSE` otherwise.

The first nine rows of the array are:

Index value	Element value	Comment
0	TRUE	valid total score (no correct answers)
1	FALSE	invalid total score
2	FALSE	invalid total score
3	TRUE	valid total score (one 3-point question correct)
4	FALSE	invalid total score
5	TRUE	valid total score (one 5-point question correct)
6	TRUE	valid total score (two 3-point questions correct)
7	FALSE	invalid total score
8	TRUE	valid total score (one 3-point question and one 5-point question correct)

For example, a total score of 6 points is valid; the value of the array element at index value 6 is `TRUE`

(a) Write pseudocode to declare `CheckTotal` and to set all elements of the array to `FALSE`

All variables used must be declared.

```

DECLARE CheckTotal : ARRAY [1/0 : 9/8] OF BOOLEAN
DECLARE index : INTEGER
FOR index ← 1 TO 9
    CheckTotal CheckTotal[index] ← FALSE
NEXT index
  
```

[4]



- (b) The pseudocode represents an algorithm to set the appropriate elements of the array to TRUE

Complete the pseudocode:

```

DECLARE EasyQ, HardQ : INTEGER
FOR EasyQ ← 0 3 3 TO 15 STEP 3 3
    FOR HardQ ← 0 TO 5 20 STEP 5 5
        CheckTotal[index index] ← TRUE
    NEXT HardQ
NEXT EasyQ
  
```

[5]

- (c) The array elements have been assigned the required values.

A module `ValidateScore()` will take an integer value representing a total score as a parameter. It will return TRUE if the total score is valid, or FALSE if it is not valid.

Describe the algorithm for `ValidateScore()` using **four** steps.

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

- Step 1 Total score is divided into two, to
separate score from easy questions and hard questions.
- Step 2 Then score from each type is
divided by 3 or 5 according to type.
- Step 3 The total number of questions and hard questions
answered correctly is known.
- Step 4 The TRUE and FALSE values are compared
and counted to match the number of total correct answers. If match is correct, return
TRUE, otherwise return FALSE (its a function).
ValidateScore()

[4]



- 5 A program is developed to satisfy a specific customer requirement.

The project follows a program development life cycle model. This model divides the development process into several different stages.

- (a) The table lists some of the development activities.

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

Activity	Name of life cycle stage
a structure chart is produced	Design ✓
a program is modified to allow it to run on new hardware	Maintenance ✓
the programmer identifies the customer's requirements	Analysis ✓
an Integrated Development Environment (IDE) provides context-sensitive help such as 'auto-complete'	Coding ✓

[4]

- (b) Alpha and beta testing has been completed. The final testing stage is carried out by the customer.

Identify this final testing stage.

..... Acceptance testing ✓ [1]





BLANK PAGE

SEEN



6 A string function Compare () will compare two strings.

The function will:

- take **four** parameters:
 - two strings, String1 and String2
 - a character, Position, to indicate whether String1 will be compared with the start ('s'), or the end ('e') of String2
 - a Boolean, CaseMatters, to indicate whether an upper case character and lower case character (for example, 'A' and 'a') are regarded as different (TRUE), or as the same (FALSE)
- return FALSE if String2 has fewer characters than String1
- return TRUE if the comparison is true, otherwise return FALSE

For example:

Parameter				Return value
String1	String2	CaseMatters	Position	
"Cat"	"Catalogue"	TRUE	's'	TRUE
"CAT"	"Catalogue"	TRUE	's'	FALSE
"CAT"	"Catalogue"	FALSE	's'	TRUE
"Cat"	"Catalogue"	TRUE	'e'	FALSE
"GUE"	"Catalogue"	FALSE	'e'	TRUE
"Catalogue"	"Cat"	TRUE	's'	FALSE

FUNCTION Compare (String1, String2 : STRING,



Write pseudocode for the function Compare()

FUNCTION Compare (CaseHtr : BOOLEAN, Pos : CHAR, Str1 : STRING,
Str2 : STRING)

DECLARE Extract1, Extract2 : STRING

RETURNS BOOLEAN

IF LENGTH(Str1) > LENGTH(Str2) THEN

RETURN FALSE

ENDIF

IF Pos = 's' THEN

Extract1 ← LEFT(Str1, 3)

Extract2 ← LEFT(Str2, 3)

ENDIF

IF Pos = 'e' THEN

Extract1 ← RIGHT(Str1, 3)

Extract2 ← RIGHT(Str2, 3)

ENDIF

IF ~~CaseHtr~~ CaseHtr = TRUE AND Extract1 = Extract2 THEN

RETURN TRUE

ENDIF

IF CaseHtr = FALSE THEN

Extract1 ← TO_LOWER(Extract1)

Extract2 ← TO_LOWER(Extract2)

~~Extract1~~

IF Extract1 = Extract2 THEN

RETURN TRUE

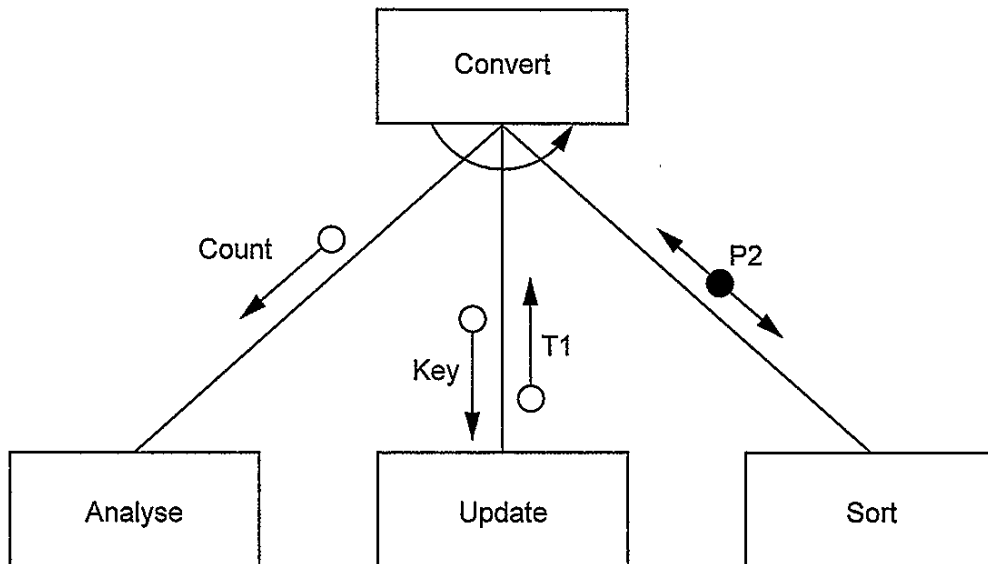
ENDIF

ENDIF

ENDFUNCTION



7 The structure chart shows part of a program design:



(a) Explain the meaning of the curved arrow symbol in this structure chart.

That indicates repetition until a certain condition is met. The module Convert will keep on calling other three modules until that condition is met. NE [2]

(b) The program designer has noted that:

- Count is the number of elements in an array
- T1 is a value representing the number of elements in an array that have been modified
- Key is of type STRING

Write the pseudocode module headers for analyse, update and sort.

Analyse:

~~PROCEDURE~~
 PROCEDURE Analyse (Count : INTEGER)

Update:

FUNCTION Update (Key : STRING) RETURNS INTEGER

Sort:

PROCEDURE Sort (BYREF P2 : BOOLEAN)





BLANK PAGE.

SEEN



8 A program is being developed to manage student book loans from a college library.

The programmer has defined a record type to define each loan.

The data items are:

Data item	Data type	Comment
StudentID	STRING	the unique ID of the student who has borrowed the book
BookID	STRING	the unique ID of the book being borrowed
OnLoan	BOOLEAN	TRUE if the book has not been returned

The programmer has defined a global array Loan to store 7000 loan records.

There are more elements in the array than books in the library. Unused elements have the StudentID set to an empty string. These may occur anywhere in the array.

The programmer has defined a program module:

Module	Description
CountLoans ()	<ul style="list-style-type: none"> called with a parameter of type STRING representing a StudentID counts the number of books currently on loan to the specified student counts the number of books that the student has already returned output both counts together with a suitable message

CountR
CountL
TempRecord
index

parameter SearchID : STRING
TempRecord ← Loan [index] FOR loop

[index]

```

IF TempRecord.StudentID = SearchID THEN
  IF TempRecord.OnLoan = TRUE THEN
    CountL ← CountL + 1
  ENDF
  IF TempRecord.OnLoan = FALSE THEN
    CountR ← CountR + 1
  ENDF
ENDIF

```

ENDIF

OUTPUT Blah Blah

ROUGH WORK





(a) Write pseudocode for module CountLoans()

```

PROCEDURE CountLoans(SearchID: STRING)
  DECLARE Rcount, Lcount, index: INTEGER
  Rcount ← 0
  Lcount ← 0
  FOR index ← 1 TO 7000
    IF Loan[index].StudentID = SearchID THEN
      IF Loan[index].OnLoan = TRUE THEN
        Lcount ← Lcount + 1
      ENDIF
      IF Loan[index].OnLoan = FALSE THEN
        Rcount ← Rcount + 1
      ENDIF
    ENDIF
  NEXT index

  OUTPUT "Student has returned", Rcount, "books" and
  "has", Lcount, "books to return."

ENDPROCEDURE

```

[7]



- (b) As a reminder, a global array Loan stores 7000 loan records with data items for each loan record:

Data item	Data type	Comment
StudentID	STRING	the unique ID of the student who has borrowed the book
BookID	STRING	the unique ID of the book being borrowed
OnLoan	BOOLEAN	TRUE if the book has not been returned

A new module is defined:

Module	Description
NewLoan()	<ul style="list-style-type: none"> called with two parameters of type STRING representing a StudentID and a BookID <i>parameters</i> searches the array for an unused loan record <i>search</i> <i>if found</i>, updates the loan record and returns TRUE, <i>otherwise</i> returns FALSE <i>Return</i>

*index
empty
result*

*result ← FALSE
empty ← FALSE
index ← 1*

WHILE Loan[index].StudentID <> "" OR index <> 7000

index ← index + 1

ENDWHILE

IF Loan[index].StudentID = "" THEN

Loan[index].StudentID ← StdID

Loan[index].BookID ← BkID

Loan[index].OnLoan ← TRUE

result ← TRUE

ENDIF

RETURN result

ROUGH WORK





Write efficient pseudocode for NewLoan()

```
FUNCTION NewLoan(StdID, BKID : STRING)
    RETURNS BOOLEAN

    DECLARE index : INTEGER
    DECLARE result : BOOLEAN

    index ← 1
    result ← FALSE

    WHILE Loan[index].StudentID <> "" OR index <> 7000
        index ← index + 1
    ENDWHILE

    IF Loan[index].StudentID = "" THEN
        Loan[index].StudentID ← StdID
        Loan[index].BookID ← BKID
        Loan[index].OnLoan ← TRUE
        result ← TRUE
    ENDIF

    RETURN result

ENDFUNCTION
```

[7]



- (c) When a book is returned, the loan record will have its OnLoan data set to FALSE.

A new procedure Archive() will mark these records as unused after first saving them for future reference.

Explain how these records can be saved for future reference.

Make a new ID array. When a book is returned, ~~not~~ update its OnLoan and ~~student~~ ID, save that record's index in the new array.

[2]

- (d) It is decided to extend the program so that a new module Reminder() will send an email to the student three days before the book is due to be returned.

Outline the changes that will need to be made to the data stored and how this data will be used to generate the reminder email.

Data The student's ^{address} email will be needed, the due date for returning the book will be set, and student's name will be needed to personalize the email.

Use a check will be used to check days remaining until due date, when it's three, a personalized email is generated and sent to the student to remind about returning the book.

[2]





BLANK PAGE

SEEN





BLANK PAGE

SEEN

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.

