



Cambridge International AS & A Level

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

May/June 2021

2 hours

You must answer on the question paper.

No additional materials are needed.

INSTRUCTIONS

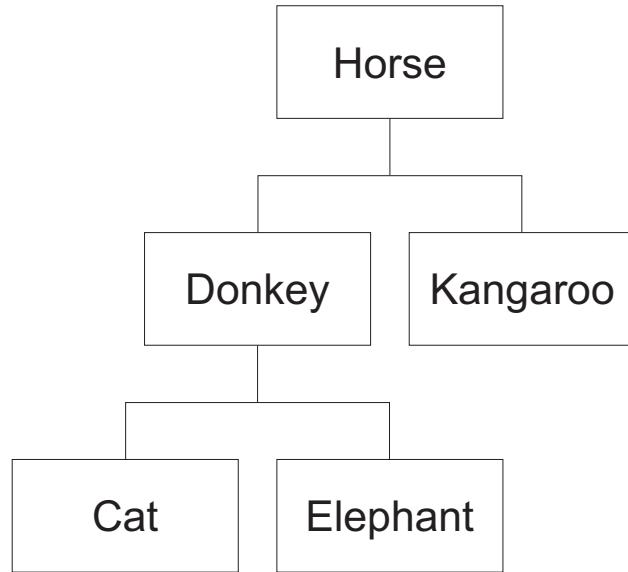
- Answer **all** questions.
- Use a black or dark blue pen.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- 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.

This document has **20** pages. Any blank pages are indicated.

- 1 An ordered binary tree stores the following data:



- (a) Identify the data in the root node of the binary tree.

..... [1]

- (b) Identify the data in **one** leaf node from the binary tree.

..... [1]

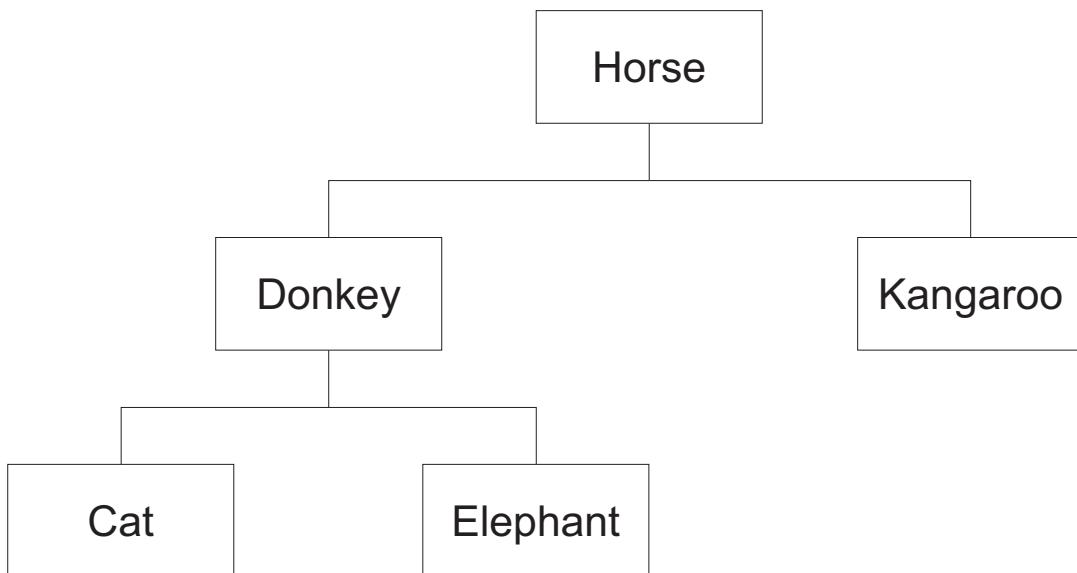
- (c) Complete the binary tree by adding the following data in the order given:

Fish

Iguana

Rabbit

Jaguar



[2]

- (d) Explain how an algorithm will search the binary tree to find **Elephant**.

.....
.....
.....
.....
..... [2]

- 2 A company stores bookings in a random file.

For each booking, the booking ID, customer ID, item ID and quantity are stored. These four values are all integers.

- (a) Write the **pseudocode** record declaration for the data type Booking.

.....
.....
.....
.....
.....
..... [2]

- (b) Each booking ID is a value between 100 000 and 999 999 inclusive.

The hash value is calculated by dividing the booking ID by 100 000 and adding 3 to the remainder.

- (i) The function `Hash()` takes the booking ID as a parameter, calculates and returns the hash value.

Write **program code** for the function `Hash()`.

Programming language

Program code

.....
.....
.....
.....
.....
..... [2]

- (ii) Calculate the hash value for each booking ID in the table.

Booking ID	Hash value
5 012 345	
8 212 350	

[1]

- (c) The function `StoreBooking()` takes a record as a parameter and stores it in the random file `TheBookings.dat`. The function uses `Hash()` to calculate the hash value for that record. The record is only stored if the value at the hashed value is `NULL`. `FALSE` is returned if there is already a record in that location and `TRUE` otherwise.

You can assume that the file exists.

Write pseudocode for the function `StoreBooking()`.

[7]

- (d) Explain how exception handling can be used when reading from a file.

.....

.....

.....

.....

[2]

- 3 Ejaz is creating a program that will allow the user to create quizzes. He is using object-oriented programming (OOP).

There are two classes: QuestionClass and QuizClass.

The class attributes and methods are in the following tables. All attributes are declared as private.

QuestionClass	
Question : STRING	// stores the question
Answer : STRING	// stores the correct answer
Difficulty : INTEGER	// stores the difficulty as an integer // from 0(easy) to 10(hard)
Constructor(QuestionP, AnswerP, DifficultyP)	// creates an instance of QuestionClass // sets the attributes to the parameter // values
GetQuestion()	// returns the question
GetDifficulty()	// returns the difficulty level
GetAnswer()	// returns the answer

QuizClass	
Questions : ARRAY[0:19] OF QuestionClass	// stores maximum 20 questions of // type QuestionClass
NumberOfQuestions : INTEGER	// stores the number of questions // in this quiz
Constructor()	// creates an instance of // QuizClass // initialises NumberOfQuestions // to 0
AddQuestion()	// adds the parameter question to // the array // increments NumberOfQuestions
GetQuestion()	// returns the next question to be // asked
CheckAnswer()	// takes an answer as a parameter // and returns TRUE if correct

- (a) Write **program code** to define the class QuizClass. You are only required to write code for the attribute declarations and constructor.

If you are writing in Python, include attribute declarations using comments.

Use your programming language's constructor method.

Programming language

Program code

.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [4]

- (b) The QuizClass method AddQuestion() takes a question object as a parameter and stores it in the next available location in the array Questions. It returns TRUE if it is successfully stored, and FALSE otherwise.

Write program code for the method AddQuestion().

Programming language

Program code

[4]

- (c) The first quiz is created with the identifier FirstQuiz.

The first question in this quiz is: "What is $100 / 5$?".

The answer is “20” and the difficulty level is 1.

Write program code to:

- declare an instance of QuizClass with the identifier FirstQuiz
 - declare an instance of QuestionClass with the identifier Question1
 - add Question1 to the array in FirstQuiz using AddQuestion().

Programming language

Program code

.....

.....
.....
.....
.....
..... [5]

- (d) The object `FirstQuiz` contains objects of type `QuestionClass`.

State the name of this OOP feature.

..... [1]

- (e) Ejaz can use an interpreter and a compiler to translate program code during the development process. The program will be distributed without any access to the source code.

- (i) State when Ejaz should use an interpreter and a compiler. Each answer must be different.

Interpreter

Compiler

[2]

- (ii) Give the name of **two** facilities that Ejaz can use to debug his program.

1

2

[2]

- (iii) Describe **one** feature of an editor that Ejaz can use when writing the program.

.....
.....
.....
..... [2]

- 4 Zara is writing a program to simulate a circular queue.

The queue, `MyNumbers`, has 10 elements. `Enqueue()` takes a parameter value and stores it at the tail of the queue. `Dequeue()` returns the item at the head of the queue.

The current state of the circular queue is:

Index	0	1	2	3	4	5	6	7	8	9
Data			31	45	89	500	23	2		

`HeadIndex`: 2

`TailIndex`: 8

- (a) Show the state of the queue, `HeadIndex` and `TailIndex` after the following operations:

`Enqueue(23)`

`Enqueue(100)`

`Dequeue()`

`Dequeue()`

`Enqueue(50)`

Index	0	1	2	3	4	5	6	7	8	9
Data										

`HeadIndex`:

`TailIndex`:

[3]

- (b) The global array, MyNumbers, is used to store the positive integer numbers for the queue.

The following global variables are used:

- HeadIndex stores the index of the first element in the queue
- TailIndex stores the index of the next free space in the queue
- NumberInQueue stores the number of items in the queue.

- (i) The function Enqueue () takes the value to be added to the queue as a parameter. The function returns TRUE if the item was added, or FALSE if the queue is full.

Complete the **pseudocode** for the function Enqueue () .

```

FUNCTION Enqueue (BYVALUE DataToInsert : INTEGER) RETURNS BOOLEAN

IF NumberInQueue > .....
THEN
RETURN FALSE
ELSE
MyNumbers [ ..... ] ← .....
TailIndex ← .....
IF TailIndex > 9
THEN
TailIndex ← .....
ENDIF
NumberInQueue ← NumberInQueue + 1
RETURN TRUE
ENDIF

ENDFUNCTION

```

[5]

- (ii) The function `Dequeue()` returns the value at the head of the queue, or `-1` if the queue is empty.

Complete the **pseudocode** for the function `Dequeue()`.

FUNCTION Dequeue() RETURNS INTEGER

ENDFUNCTION

[5]

- 5 The following procedure performs an insertion sort on the global array TheArray that has 10 elements.

Complete the pseudocode for the procedure InsertionSort().

```

PROCEDURE InsertionSort()

    DECLARE Count : INTEGER
    DECLARE Counter : INTEGER
    DECLARE Temp : INTEGER

    Count ← .....
    WHILE Count < 10
        Temp ← TheArray[Count]
        Counter ← Count .....
        WHILE ..... >= 0 AND TheArray[Counter] > .....
            TheArray[Counter + 1] ← TheArray[Counter]
            Counter ← Counter - 1
        ENDWHILE
        TheArray[.....] ← Temp
        Count ← Count + 1
    ENDWHILE
ENDPROCEDURE

```

[5]

- 6 A social networking website only allows people who are over 16 years old to join.

To create an account, the user must enter:

- their age
- a unique username, which is compared to others in the database
- a password that must be at least 8 characters long, with at least one upper case letter, one lower case letter, one symbol and one digit.

If the user is not old enough to join the network, the statement “Too young” is displayed.

If the user is old enough, but the username is already taken, the statement “Choose another username” is displayed.

If the user is old enough, but the password does not meet the requirements, the statement “Password does not meet requirements” is displayed.

- (a) Complete the decision table for the social networking website.

Conditions	Available username	N	Y	N	Y	N	Y	N	Y
	Suitable password	N	N	Y	Y	N	N	Y	Y
	Age > 16	N	N	N	N	Y	Y	Y	Y
Actions	“Too young”								
	“Choose another username”								
	“Password does not meet requirements”								

[4]

- (b) Simplify the decision table by removing the redundancies.

Conditions	Available username								
	Suitable password								
	Age > 16								
Actions	“Too young”								
	“Choose another username”								
	“Password does not meet requirements”								

[3]

- 7 Anika is designing a computer game. The user controls a character that moves around a virtual world.

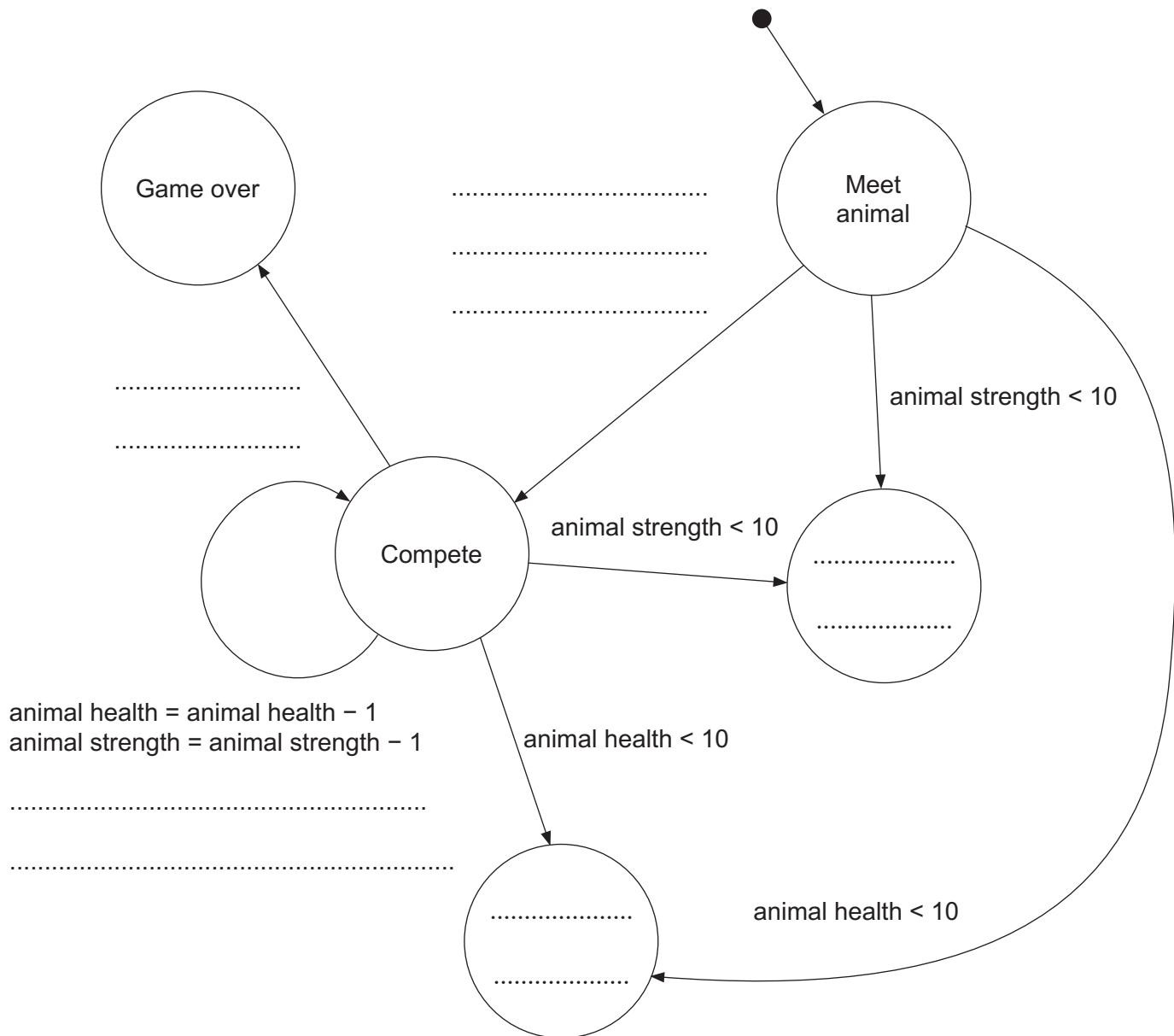
When the character meets an animal:

- if the animal's strength is less than 10, the animal runs away
- if the animal's health is less than 10, it is caught by the character
- if the animal's strength and health are both 10 or more, the character and the animal compete.

When the character and animal compete, the animal's health, animal's strength and character's health are decreased by 1. This is repeated until one of the following conditions is met:

- the character's health goes to 0, the game is over
- the animal's strength goes below 10, the animal runs away
- the animal's health goes below 10, the animal is caught.

Complete the state-transition diagram for this part of the program.



[5]

- 8 The table shows assembly language instructions for a processor that has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

	Instruction		Explanation
Label	Op code	Operand	
	LDM	#n	Immediate addressing. Load the number n to ACC
	LDI	<address>	Direct addressing. Load the contents of the location at the given address to ACC
	LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC
	LDR	#n	Immediate addressing. Load the number n to IX
	STO	<address>	Store the contents of ACC at the given address
	ADD	<address>	Add the contents of the given address to ACC
	INC	<register>	Add 1 to the contents of the register (ACC or IX)
	CMP	#n	Compare the contents of ACC with number n
	JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
	LSL	#n	Bits in ACC are shifted n places to the left. Zeroes are introduced on the right hand end
	LSR	#n	Bits in ACC are shifted n places to the right. Zeroes are introduced on the left hand end
	OUT		Output to the screen the character whose ASCII value is stored in ACC
	END		Return control to the operating system
<label>:	<op code>	<operand>	Labels an instruction
<label>:	<data>		Gives a symbolic address <label> to the memory location with contents <data>

An algorithm stores a 3-character word. It takes each character in turn, multiplies its value by 2 and outputs the new character.

Complete the following assembly language program for the algorithm using the instruction set provided on the previous page.

Instruction			Comment
Label	Op code	Operand	
			// initialise Index Register to 0
	LDX	character	// load character and multiply by 2
	OUT		// output the new character
	INC	IX	// increment the Index Register
	LDD	count	// loop 3 times
	STO	count	
	CMP	#3	
		LOOP	
	END		// end program
count:	0		
character:	B01000001		// the 3-character stored word
	B10001110		
	B01000100		

[5]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

May/June 2016

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **19** printed pages and **1** blank page.

- 1 A linked list abstract data type (ADT) is to be used to store and organise surnames.

This will be implemented with a 1D array and a start pointer. Elements of the array consist of a user-defined type. The user-defined type consists of a data value and a link pointer.

Identifier	Data type	Description
LinkedList	RECORD	User-defined type
Surname	STRING	Surname string
Ptr	INTEGER	Link pointers for the linked list

- (a) (i) Write **pseudocode** to declare the type `LinkedList`.

.....

[3]

- (ii) The 1D array is implemented with an array `SurnameList` of type `LinkedList`.

Write the **pseudocode** declaration statement for `SurnameList`. The lower and upper bounds of the array are 1 and 5000 respectively.

..... [2]

- (b) The following surnames are organised as a linked list with a start pointer `StartPtr`.

`StartPtr: 3`

	1	2	3	4	5	6	5000
Surname	Liu	Yang	Chan	Wu	Zhao	Huang	...	
Ptr	4	5	6	2	0	1	...	

State the value of the following:

- (i) `SurnameList[4].Surname` [1]

- (ii) `SurnameList[StartPtr].Ptr` [1]

- (c) Pseudocode is to be written to search the linked list for a surname input by the user.

Identifier	Data type	Description
ThisSurname	STRING	The surname to search for
Current	INTEGER	Index to array SurnameList
StartPtr	INTEGER	Index to array SurnameList. Points to the element at the start of the linked list

- (i) Study the pseudocode in part (c)(ii).

Complete the table above by adding the missing identifier details.

[2]

- (ii) Complete the pseudocode.

```

01 Current ← .....
02 IF Current = 0
03   THEN
04     OUTPUT .....
05 ELSE
06   IsFound ← .....
07   INPUT ThisSurname
08 REPEAT
09   IF ..... = ThisSurname
10     THEN
11       IsFound ← TRUE
12       OUTPUT "Surname found at position ", Current
13     ELSE
14       // move to the next list item
15       .....
16   ENDIF
17 UNTIL IsFound = TRUE OR .....
18 IF IsFound = FALSE
19   THEN
20   OUTPUT "Not Found"
21 ENDIF
22 ENDIF

```

[6]

- 2 (a) (i) State what is meant by a recursively defined procedure.

.....
.....

[1]

- (ii) Write the line number from the pseudocode shown in part (b) that shows the procedure X is recursive.

[1]

- (b) The recursive procedure X is defined as follows:

```

01 PROCEDURE X(Index, Item)
02     IF MyList[Index] > 0
03         THEN
04             IF MyList[Index] >= Item
05                 THEN
06                     MyList[Index] ← MyList[Index + 1]
07                 ENDIF
08             CALL X(Index + 1, Item)
09         ENDIF
10     ENDPROCEDURE

```

An array MyList is used to store a sorted data set of non-zero integers. Unused cells contain zero.

	1	2	3	4	5	6	7	8	9	10
MyList	3	5	8	9	13	16	27	0	0	0

- (i) Complete the trace table for the dry-run of the pseudocode for the procedure
CALL X(1, 9).

		MyList									
Index	Item	1	2	3	4	5	6	7	8	9	10
1	9	3	5	8	9	13	16	27	0	0	0

[4]

- (ii) State the purpose of procedure X when used with the array MyList.

.....
.....

[1]

- 3 A car hire company hires cars to customers. Each time a car is hired, this is treated as a transaction.

For each transaction, the following data are stored.

For the customer:

- customer name
- ID number

For the hire:

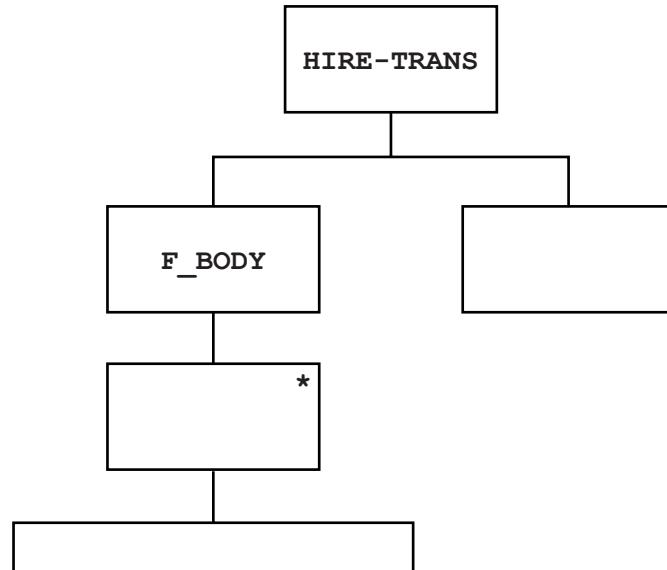
- car registration
- hire start date
- number of days hired

The transaction data are stored in a text file HIRE-TRANS. The file is made up of a file body, F_BODY, and a file trailer, F_TRAILER.

F_BODY has one transaction, TRANS, on each line.

- (a) The first step in Jackson Structured Programming (JSP) design is to produce a JSP data structure diagram.

Complete the following JSP data structure diagram.



- (b) The computer system will produce many printed reports.

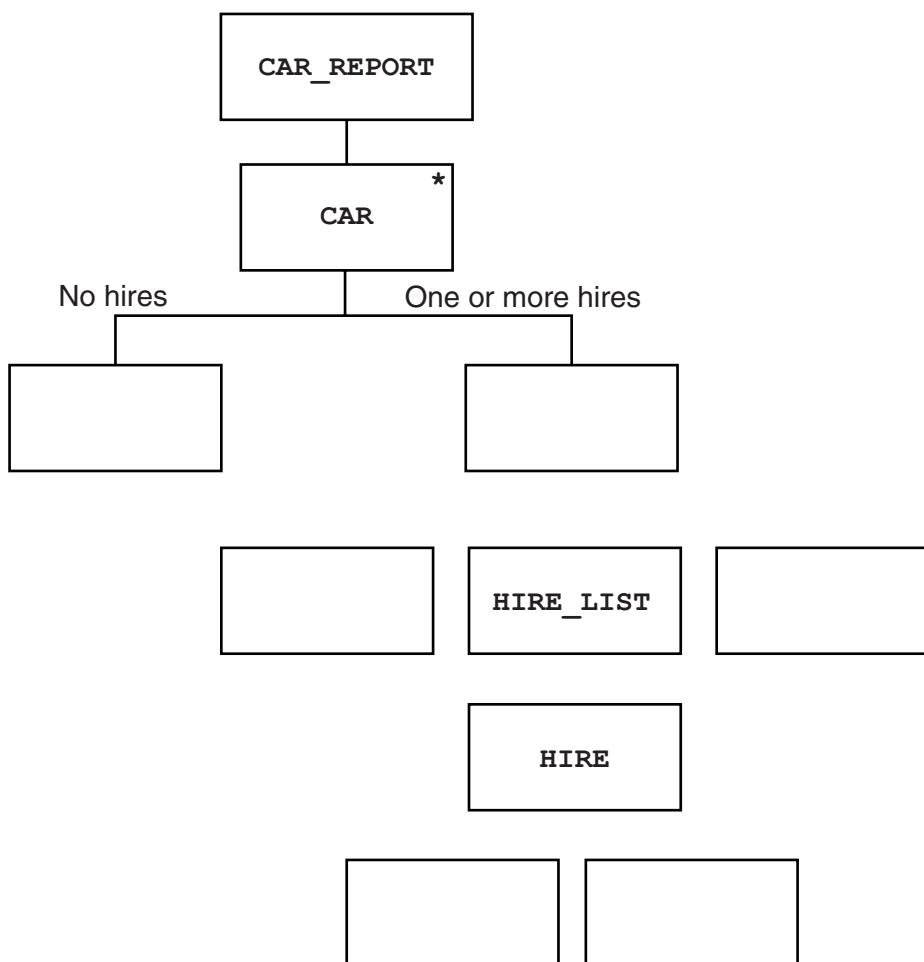
One report is CAR_REPORT. This displays all hire data for all cars.

For each car, the following data are displayed:

- the car data
- a list of all the hires
- the total number of hires

A car with zero hires is not included on the report.

Complete the following CAR_REPORT JSP data structure diagram.



[5]

- 4 When a car reaches a certain age, a safety assessment has to be carried out. A car's brakes and tyres must be tested. The tyre test result and the brakes test result for each car are recorded. If the car passes the assessment, a safety certificate is issued.

Cars have a unique three-character registration.

The following knowledge base is used:

```

01 car(a05).
02 car(h04).
03 car(a03).
04 car(h07).
05 car(a23).
06 car(p05).
07 car(b04).
08 carRegYear(a05, 2015).
09 carRegYear(h04, 2013).
10 carRegYear(a03, 2008).
11 carRegYear(h07, 2011).
12 carRegYear(a23, 2008).
13 carRegYear(p05, 2014).
14 carRegYear(b04, 2014).
15 testBrakes(h07, pass).
16 testTyres(h07, fail).
17 testBrakes(a03, fail).
18 testTyres(a03, fail).
19 testBrakes(a23, pass).
20 testTyres(a23, pass).
21 carAssessmentDue if carRegYear(Car, RegYear)
                           and RegYear <= DeadlineYear.
22 issueCertificate(Car) if testTyres(Car, Result) and
                           testBrakes(Car, Result) and Result = pass.

```

- (a) (i) DeadlineYear is assigned value 2011.

Identify the car registrations for cars which are due to be tested.

..... [1]

- (ii) State how clause 22 determines whether or not a safety certificate will be issued.

..... [1]

- (b)** If a car fails one of the two tests, a retest is allowed.

Write a new rule for this.

retestAllowed(.....) if

.....

..... [3]

- (c)** Logic programming uses a data structure called a list.

A new fact is added to the knowledge base.

23 carList = [a03, p05, b04, h04, h07, a23].

The following notation and operators are to be used with a list:

[X | Y] denotes a list with:

- X the first list element
- Y the list consisting of the remaining list elements

[] denotes an empty list

- (i)** The list [a07, p03] is denoted by [A | B]

State the value of A and B.

A =

B = [2]

- (ii)** The lists [c03, d02, n05 | C] and [c03, d02, n05, p05, m04] are identical.

State the value of C.

C = [1]

- (iii)** The list [a06, a02] is denoted by [D, E | F]

State the value of F.

F = [1]

- (d) The predicate `conCatCompare` is defined as a rule and returns TRUE or FALSE as follows:

`conCatCompare(X, Y, Z)`

Concatenates the lists X and Y and compares the new list with list Z.

If equal, the clause evaluates to TRUE, otherwise FALSE.

Consider the clause:

`conCatCompare(X, Y, [a7,b6,c4])`

If:

- the clause evaluates to TRUE
- and Y represents the list [a7, b6, c4]

State the value of X.

X = [1]

- 5 (a) A program calculates the exam grade awarded from a mark input by the user. The code is written as a function CalculateGrade.

The function:

- has a single parameter **Mark** of **INTEGER** data type
- returns the grade awarded **Grade** of **STRING** data type

The logic for calculating the grade is as follows:

Mark	Grade
Under 40	FAIL
40 and over and under 55	PASS
55 and over and under 70	MERIT
70 and over	DISTINCTION

The programmer designs the following table for test data:

Mark	Description	Expected result (Grade)
	Normal	
	Abnormal	
	Extreme/Boundary	

- (i) Complete the table above. [3]
- (ii) State why this table design is suitable for black box testing.

..... [1]

(b) When designing and writing program code, explain what is meant by:

- an exception
- exception handling

.....
.....
.....
.....
.....
.....

[3]

(c) A program is to be written to read a list of exam marks from an existing text file into a 1D array.

Each line of the file stores the mark for one student.

State **three** exceptions that a programmer should anticipate for this program.

1

.....

2

.....

3

.....

[3]

- (d) The following pseudocode is to read two numbers:

```

01  DECLARE Num1      : INTEGER
02  DECLARE Num2      : INTEGER
03  DECLARE Answer : INTEGER
04  TRY
05      OUTPUT "First number..."
06      INPUT Num1
07      OUTPUT "Second number..."
08      INPUT Num2
09      Answer ← Num1 / (Num2 - 6)
10      OUTPUT Answer
11  EXCEPT ThisException : EXCEPTION
12      OUTPUT ThisException.Message
13  FINALLY
14      // remainder of the program follows
...
29
30 ENDTRY

```



The programmer writes the corresponding program code.

A user inputs the number 53 followed by 6. The following output is produced:

```

First number...53
Second number...6
Arithmetic operation resulted in an overflow

```

- (i) State the pseudocode line number which causes the exception to be raised.

.....

[1]

- (ii) Explain the purpose of the pseudocode on lines 11 and 12.

.....

[3]

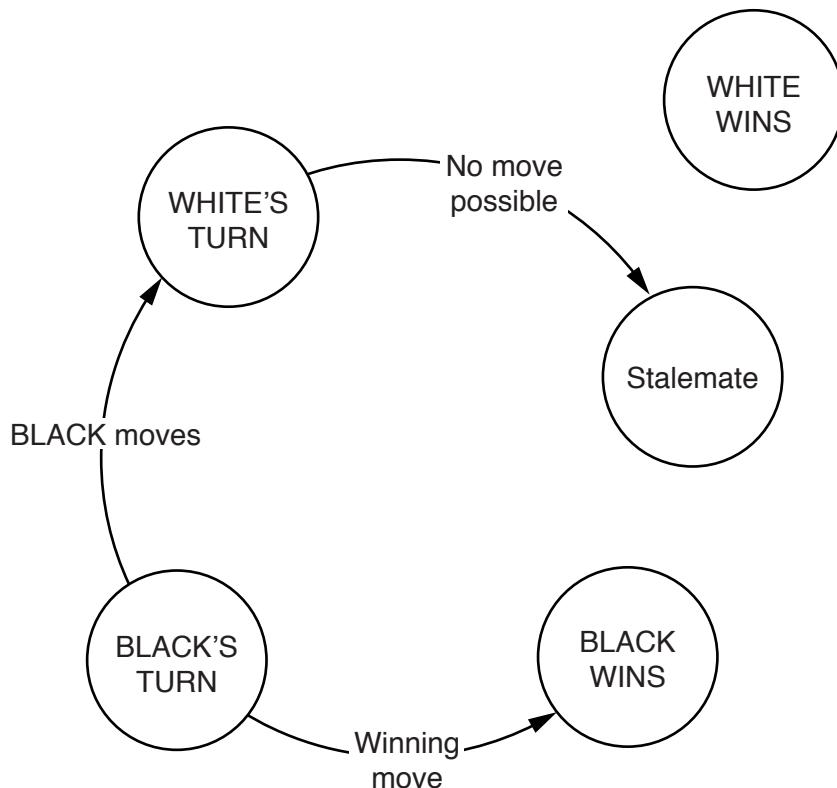
- 6 In a board game, one player has white pieces and the other player has black pieces. Players take alternate turns to move one of their pieces. White always makes the first move.

The game ends if:

- a player is unable to make a move when it is their turn. In this case, there is no winner. This is called 'stalemate'.
- a player wins the game as a result of their last move and is called a 'winner'.

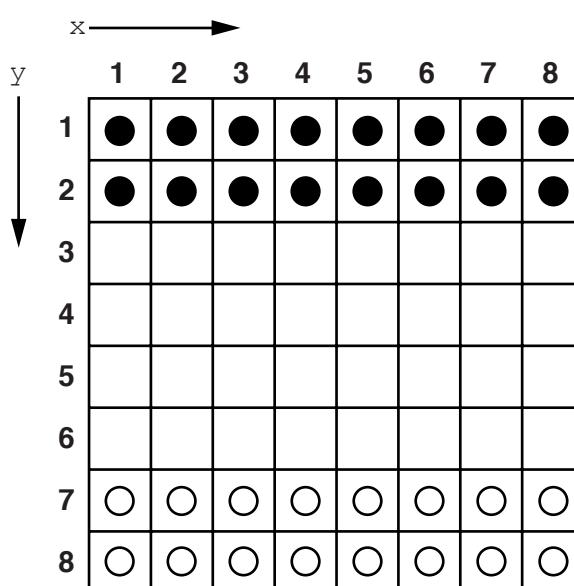
- (a) A state-transition diagram is drawn to clarify how the game is played.

Complete the following state-transition diagram.



[4]

- (b) The layout of the board at the start of the game is shown below:



The programmer decides to use a 2D array to represent the board. The index numbering to be used is as shown.

Each square on the board is either occupied by one piece only, or is empty.

The data stored in the array indicate whether or not that square is occupied, and if so, with a black piece or a white piece.

- (i) Write **program code** to initialise the contents of the array to represent the board at the start of the game. Use characters as follows for each square:

- 'B' represents a black piece 
 - 'W' represents a white piece 
 - 'E' represents an empty square

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language

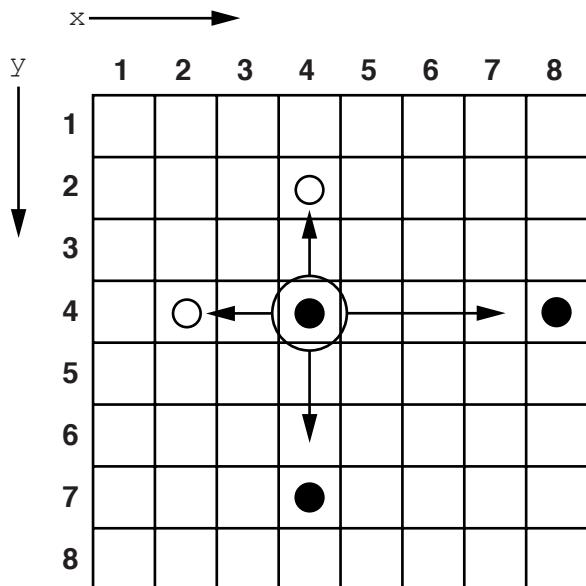
- (ii) When a piece is to be moved, a procedure will calculate and output the possible destination squares for the moving piece.

A piece can move one or more squares, in the x or y direction, from its current position.

This will be a move:

- either to an empty square, with no occupied squares on the way
- or to a square containing a piece belonging to another player, with no occupied squares on the way. The other player's piece is then removed.

For example, for the circled black piece there are nine possible destination squares. Each of the two destination squares contains a white piece which would be removed.



The program requires a procedure `ValidMoves`.

It needs three parameters:

- `PieceColour` – colour of the moving piece
- `xCurrent` – current x position
- `yCurrent` – current y position

The procedure will calculate all possible destination squares **in the x direction only**.

Example output for the circled black piece is:

```

Possible moves are:
Moving LEFT
3 4
2 4 REMOVE piece
Moving RIGHT
5 4
6 4
7 4
  
```

Write **program code** for procedure `ValidMoves` with the following procedure header:

```

PROCEDURE ValidMoves(PieceColour : CHAR, xCurrent : INTEGER,
                      yCurrent : INTEGER).
  
```

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language

. [5]

- (c) The problem is well suited to an object-oriented design followed by object-oriented programming.

- (i) Describe how classes and objects could be used in this problem.

.....
.....
.....
.....
.....

[2]

- (ii) For a class you identified in part(c)(i), state **two** properties and **two** methods.

Class

Properties

1
2

Methods

1,
2

[2]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

October/November 2015

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **15** printed pages and **1** blank page.

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

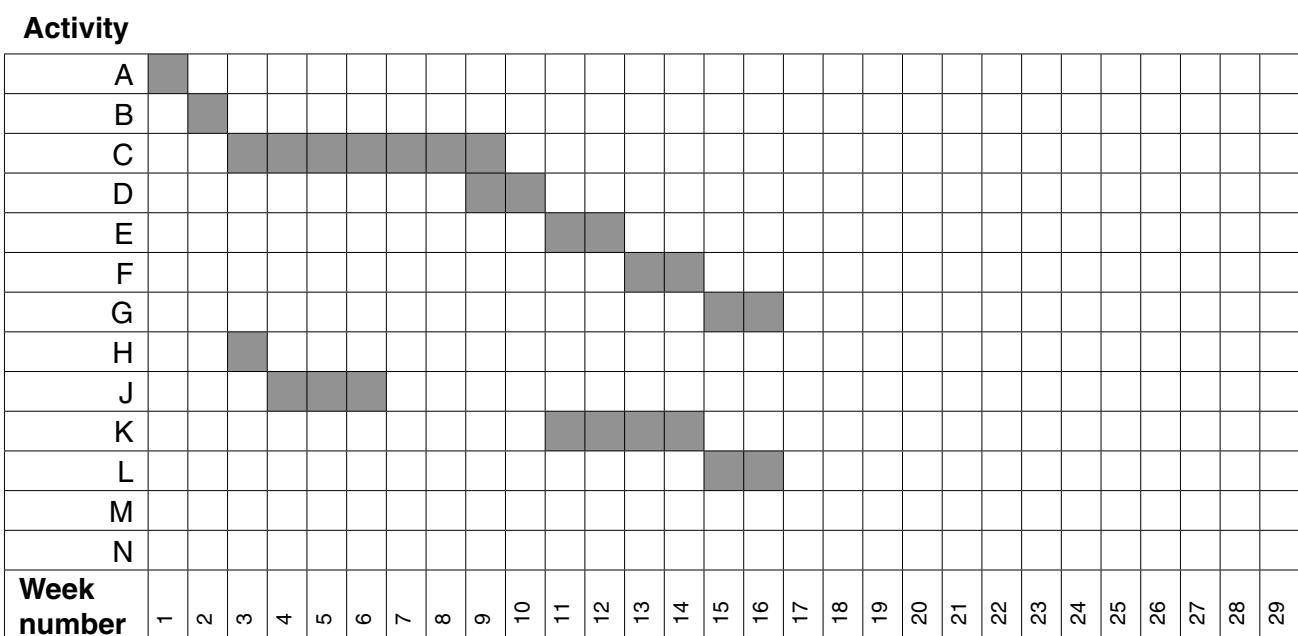
Complete the statement to indicate which high-level programming language you will use.

Programming language

- 1 A large software house has been asked to supply a computerised solution for a business. The project manager has drawn up a list of activities and their likely duration.

Activity	Description	Weeks to complete
A	Write requirement specification	1
B	Produce program design	1
C	Write module code	7
D	Module testing	2
E	Integration testing	2
F	Alpha testing	2
G	Install software and carry out acceptance testing	2
H	Research and order hardware	1
J	Install delivered hardware	3
K	Write technical documentation	4
L	Write user training guide	2
M	Train users on installed hardware and software	1
N	Sign off final system	1

- (a) From this data a GANTT chart is constructed.



- (i) Complete the GANTT chart by adding activities M and N. [2]

(ii) State the earliest completion date

Week number [1]

- (b)** There are problems with the progress of the project:

- Activity E showed that the code contained major errors. The senior programmer now estimates that:
 - further module coding will require another 2 weeks
 - further module testing will require another 2 weeks
 - further integration testing will require another 2 weeks
 - The hardware delivery is delayed by 16 weeks.

A revised GANTT chart is now required.

- (i) Complete the chart in the grid below.

[9]

- (ii) State the new estimated completion date.

Week number [1]

- 2** A declarative programming language is used to represent the following facts and rules:

```

01 male(ahmed).
02 male(raul).
03 male(ali).
04 male(phiippe).
05 female(aisha).
06 female(gina).
07 female(meena).
08 parent(ahmed, raul).
09 parent(aisha, raul).
10 parent(ahmed, phiippe).
11 parent(aisha, phiippe).
12 parent(ahmed, gina).
13 parent(aisha, gina).
14 mother(A, B) IF female(A) AND parent(A, B).

```

These clauses have the following meaning:

Clause	Explanation
01	Ahmed is male
05	Aisha is female
08	Ahmed is a parent of Raul
14	A is the mother of B if A is female and A is a parent of B

- (a)** More facts are to be included.

Ali and Meena are the parents of Ahmed.

Write the additional clauses to record this.

15

16 [2]

- (b)** Using the variable C, the goal

parent(ahmed, C)

returns

C = raul, phiippe, gina

Write the result returned by the goal

parent(P, gina)

P = [2]

- (c) Use the variable M to write the goal to find the mother of Gina.

..... [1]

- (d) Write the rule to show that F is the father of C .

$\text{father}(F, C)$

IF

..... [2]

- (e) Write the rule to show that X is a brother of Y .

$\text{brother}(X, Y)$

IF

.....

.....

..... [4]

- 3 A college has two types of student: full-time and part-time.

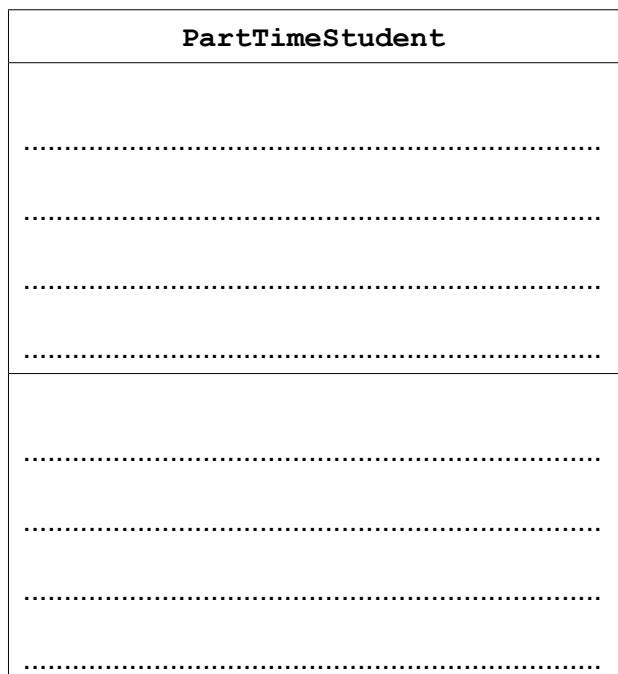
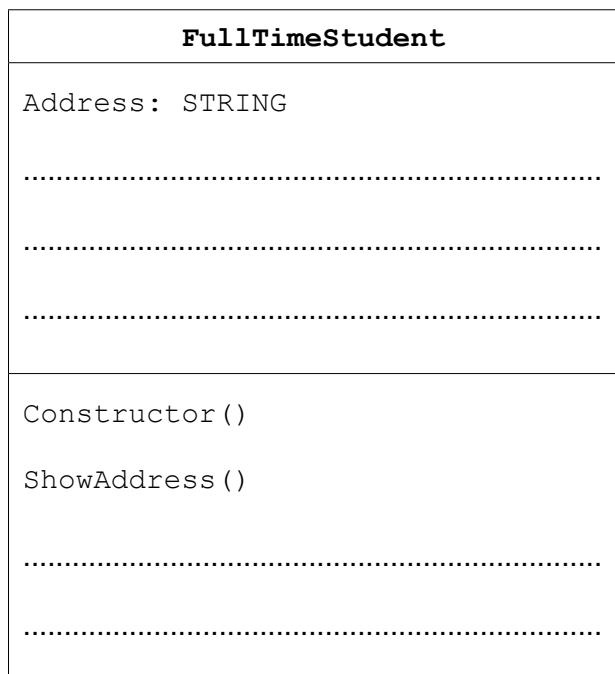
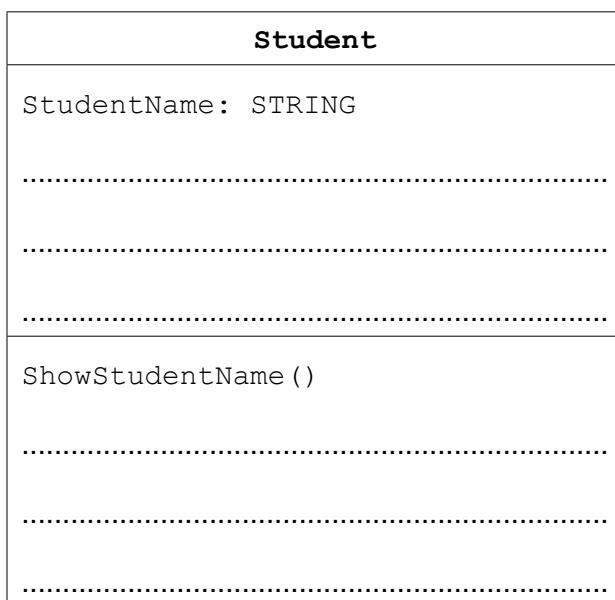
All students have their name and date of birth recorded.

A full-time student has their address and telephone number recorded.

A part-time student attends one or more courses. A fee is charged for each course. The number of courses a part-time student attends is recorded, along with the total fee and whether or not the fee has been paid.

The college needs a program to process data about its students. The program will use an object-oriented programming language.

- (a) Complete the class diagram showing the appropriate properties and methods.



(b) Write program code:

(i) for the class definition for the superclass Student.

Programming language

.....

.....

.....

.....

.....

.....

.....

.....

[2]

(ii) for the class definition for the subclass FullTimeStudent.

Programming language

.....

.....

.....

.....

.....

.....

.....

.....

[3]

(iii) to create a new instance of FullTimeStudent with:

- identifier: NewStudent
- name: A. Nyone
- date of birth: 12/11/1990
- telephone number: 099111

Programming language
.....
.....
.....
.....
..... [3]

4 A dictionary Abstract Data Type (ADT) has these associated operations:

- Create dictionary (CreateDictionary)
- Add key-value pair to dictionary (Add)
- Delete key-value pair from dictionary (Delete)
- Lookup value (Lookup)

The dictionary ADT is to be implemented as a two-dimensional array. This stores key-value pairs.

The pseudocode statement

```
DECLARE Dictionary : Array[1:2000, 1:2] OF STRING
```

reserves space for 2000 key-value pairs in array Dictionary.

The CreateDictionary operation initialises all elements of Dictionary to the empty string.

- (a) The hashing function Hash is to extract the first letter of the key and return the position of this letter in the alphabet. For example Hash("Action") will return the integer value 1.
 (Note: The ASCII code for the letter A is 65.)

Complete the pseudocode:

```
FUNCTION Hash (.....) RETURNS .....
```

```
DECLARE Number : INTEGER
```

```
Number ← .....
```

```
ENDFUNCTION
```

[5]

- (b) The algorithm for adding a new key-value pair to the dictionary is written, using pseudocode, as a procedure.

```

PROCEDURE Add(NewKey : STRING, NewValue : STRING)
    Index ← Hash(NewKey)
    Dictionary[Index, 1] ← NewKey           // store the key
    Dictionary[Index, 2] ← NewValue         // store the value
ENDPROCEDURE

```

An English-German dictionary of Computing terms is to be set up.

- (i) Dry-run the following procedure calls by writing the keys and values in the correct elements of Dictionary.

```

Add("File", "Datei")
Add("Disk", "Platte")
Add("Error", "Fehler")
Add("Computer", "Rechner")

```

Dictionary		
Index	Key	Value
1		
2		
3		
4		
5		
6		
7		
8		
:		
:		
1999		
2000		

[2]

- (ii) Another procedure call is made: Add("Drive", "Laufwerk")

Explain the problem that occurs when this key-value pair is saved.

.....

.....

.....

.....

[2]

- (iii) Describe a method to handle the problem identified in part (b)(ii).

.....
.....
.....
.....

[2]

- (iv) Write **pseudocode** to implement the method you described in part (b) (iii). Choose line numbers to indicate where your pseudocode should be inserted in the given pseudocode.

```
10  PROCEDURE Add (NewKey : STRING, NewValue : STRING)
20      Index ← Hash (NewKey)
30      Dictionary [Index, 1] ← NewKey      // store the key
40      Dictionary [Index, 2] ← NewValue    // store the value
50  ENDPROCEDURE
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

[4]

Question 5 begins on page 12.

- 5 The table shows assembly language instructions for a processor which has one general purpose register – the Accumulator (ACC).

Instruction		Explanation
Op Code	Operand	
LDM #n		Immediate addressing. Load the number n to ACC
LDD <address>		Direct addressing. Load the contents of the given address to ACC
STO <address>		Store the contents of ACC at the given address
ADD <address>		Add the contents of the given address to the ACC
INC <register>		Add 1 to the contents of the register
CMP <address>		Compare the contents of ACC with the contents of <address>
JPN <address>		Following a compare instruction, jump to <address> if the compare was False
END		Return control to the operating system

- (a) (i) Dry-run this assembly language program using the trace table.

500	LDD 512
501	ADD 509
502	STO 512
503	LDD 511
504	INC ACC
505	STO 511
506	CMP 510
507	JPN 500
508	END
509	7
510	3
511	0
512	0

Trace table

[5]

- (ii) Explain the role address 511 has in this assembly language program.

[2]

[2]

- (b) Using opcodes from the given table, write instructions to set the value at address 509 to 12.

[2]

[2]

- 6 A company keeps details of its stock items in a file of records, StockFile.

- (a) The record fields are the ProductCode, the Price and the NumberInStock.

Write the **program code** to declare the record structure StockItem.

Programming language

.....
.....
.....
.....
.....
.....
.....
.....

[4]

- (b) Before records can be read from file StockFile, the file needs to be opened.

- (i) Complete the pseudocode.

01 TRY

02 OPENFILE

03 EXCEPT

04

05 ENDTRY

[2]

- (ii) Explain the reason for including lines 01, 03, 04, 05.

.....
.....
.....
.....
.....

[2]

- (c) A stock report program uses a variable of type `StockItem` declared as follows:

```
DECLARE ThisStockItem : Stockitem
```

The program reads each record in the file `StockFile` in turn.

The program outputs the fields `ProductCode` and `NumberInStock` for each record.

Write **pseudocode** for this.

.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [4]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

October/November 2018

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **15** printed pages and **1** blank page.

- 1 A bank provides bank accounts to customers.

The following pseudocode represents the operation of the bank account.

```
CALL OpenAccount()
CALL AccountLifeTime()
CALL CloseAccount()

PROCEDURE AccountLifeTime()
    REPEAT
        CALL Transactions()
    UNTIL AccountClosed() = TRUE
ENDPROCEDURE

PROCEDURE CloseAccount()
    IF ReopenAccount() = TRUE
        THEN
            CALL FlagToReopen()
        ELSE
            CALL DeletePermanently()
    ENDIF
ENDPROCEDURE
```

- (a) Complete the JSP structure diagram for this bank account from the pseudocode given.

Bank account

[6]

- (b) A transaction can be a credit (deposit) or a debit (withdrawal).

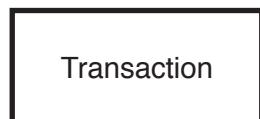
There are two types of transaction that are credits (deposits). These are:

- SWIFT payment
- BACS payment

There are three types of transaction that are debits (withdrawals). These are:

- Debit card payment
- Cheque payment
- Online payment

Complete the JSP structure diagram to represent these additional requirements.



[4]

- 2 A declarative language is used to represent facts and rules about dogs that perform tasks to help people.

```

01 type(pointer, gundog).
02 type(flushing, gundog).
03 type(retriever, gundog).
04
05 is_a(labrador, retriever).
06 is_a(newfoundland, retriever).
07 is_a(cocker_spaniel, flushing).
08 is_a(springer_spaniel, flushing).
09 is_a(king_charles, flushing).
10 is_a(english_setter, pointer).
11 is_a(irish_setter, pointer).
12
13 fav_bird(pointer, grouse).
14 fav_bird(flushing, pheasant).
15 fav_bird(retriever, waterfowl).
16
17 type(X, Y) IF is_a(Z, X) AND type(Z, Y).

```

These clauses have the following meaning:

Clause	Explanation
01	A pointer is a type of gundog.
05	A labrador is an example of a retriever.
13	The favourite bird of a pointer is a grouse.
17	X is a type of Y if Z is an example of X and Z is a type of Y.

- (a) More facts are to be included.

A **standard poodle** is an example of a waterdog. A waterdog is a type of gundog.

Write the additional clauses to record these facts.

18

19

[2]

- (b) Using the variable `P`, the goal

```
is_a(P, retriever)
```

returns

```
P = labrador, newfoundland
```

Write the result returned by the goal

```
is_a(H, pointer)
```

`H = [2]`

- (c) Write a query, using the variable `W`, to find out what an **irish setter** is an example of.

..... [2]

- (d) `Y` is the favourite bird of dog `X`.

Complete the following rule:

`fav_bird(X, Y) IF [3]`

.....

(e) State the value returned by the goal

```
NOT(is_a(labrador, retriever))
```

..... [1]

- 3 A bubble sort algorithm is used to sort an integer array, List. This algorithm can process arrays of different lengths.

(a) Write **pseudocode** to complete the bubble sort algorithm shown.

```

01 FOR Outer ← ..... TO 0 STEP - 1
02   FOR Inner ← 0 TO (.....)
03     IF ..... > .....
04     THEN
05       Temp ← .....
06       List[Inner] ← .....
07       List[Inner + 1] ← .....
08     ENDIF
09   ENDFOR
10 ENDFOR

```

[7]

(b) (i) State the order of the sorted array.

..... [1]

(ii) State which line of the algorithm you would change to sort the array into the opposite order.

State the change you would make.

Line

Change

..... [1]

- (c) Use **pseudocode** to write an alternative version of this bubble sort algorithm that will exit the algorithm when the list is fully sorted.

[4]

- 4 A circus is made up of performers. There are three types of performer: clown, acrobat and aerial.

The following data are stored for each performer.

- First name
- Last name
- Secondary role (that can be edited)
- Stage name (that can be edited)
- Type of performer (PerfType)

The following statements apply to performers.

- An acrobat may or may not use fire in his or her act.
- An aerial performer can be one of two types: either catcher or flyer.
- Each clown has an item, such as a water-spraying flower or a unicycle.
- Each clown also has a musical instrument, such as a guitar or an oboe.

Each of the three types of performer has a method that will display all of the information about that performer in a specific format. For example:

Sally Superstar (real name Sally Smith) is an acrobat. Fire is part of Sally Superstar's act. When not performing, Sally Superstar is a set changer.

- (a) Complete the following class diagram to show the **attributes**, **methods** and **inheritance** for the program.

You do not need to write the get and set methods.

Performer
FirstName : STRING
LastName : STRING
SecondaryRole : STRING
StageName : STRING
PerfType : STRING
Constructor()
EditSecondaryRole()
EditStageName()

Acrobat
UseFire : BOOLEAN
Constructor()
PerformerInfo()

Clown
.....
.....
Constructor()
.....

Aerial
.....
.....
Constructor()
.....

[4]

- (b)** Write program code for the Performer class.

Programming language

Program code

- (c) The program will display the acrobat information as follows:

Sally Superstar (real name Sally Smith) is an acrobat. Fire is part of Sally Superstar's act. When not performing, Sally Superstar is a set changer.

Write program code for the Acrobat class.

Programming language

Program code

..... [8]

- (d) Information about a performer is as follows:

Amazing Alex (real name Alex Tan) is an acrobat. Fire is part of Alex's act. When not performing, Amazing Alex is a popcorn seller.

- (i) Write **program code** to create an instance of an object with the identifier `Acrobat_1`.

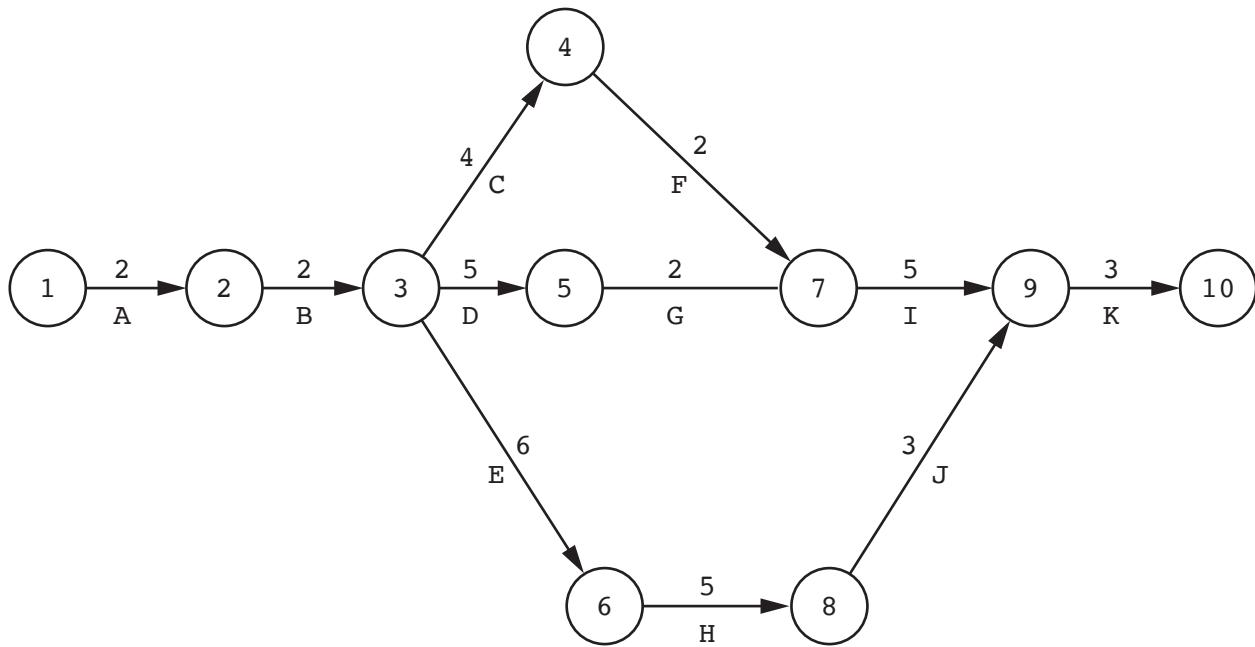
All attributes of the instance should be fully initialised.

..... [3]

- (ii) Explain **inheritance** with reference to the circus example.

..... [2]

- 5 A Program Evaluation Review Technique (PERT) chart has been constructed for a project that is at the planning stage.



- (a) Complete the following GANTT chart using the information in the PERT chart.

[5]

- (b) There are three teams working on the project. Each team is able to work on any of the activities.

Explain, with reference to the PERT chart, how work can be allocated to the three teams.

.....
.....
.....
.....
.....
.....

[2]

- (c) The PERT chart is used to calculate the critical path for the project.

- (i) List the activities that form the critical path using the given PERT chart on page 12.

..... [1]

- (ii) Explain the importance of the critical path for project delivery.

.....
.....
.....
.....
.....

[2]

- 6 A linked list abstract data type (ADT) is created. This is implemented as an array of records. The records are of type `ListElement`.

An example of a record of `ListElement` is shown in the following table.

Data Item	Value
Country	"Scotland"
Pointer	1

- (a) (i) Use **pseudocode** to write a definition for the record type, `ListElement`.

.....

[3]

- (ii) Use **pseudocode** to write an array declaration to reserve space for only 15 nodes of type `ListElement` in an array, `CountryList`. The lower bound element is 1.

..... [2]

- (b) The program stores the position of the last node in the linked list in `LastNode`. The last node always has a `Pointer` value of -1. The position of the node at the head of the list is stored in `ListHead`.

After some processing, the array and variables are in the following state.

CountryList		
	Country	Pointer
ListHead	1	
1	"Wales"	2
LastNode	3	
3	"Scotland"	4
4		-1
5	"England"	5
6	"Brazil"	6
7	"Canada"	7
8	"Mexico"	8
9	"Peru"	9
10	"China"	10
11		11
12		12
13		13
14		14
15		15
		3

A **recursive** algorithm searches the list for a value, deletes that value, and updates the required pointers. When a node value is deleted, it is set to empty "" and the node is added to the end of the list.

A node value is deleted using the pseudocode statement

```
CALL DeleteNode ("England", 1, 0)
```

Complete the following **pseudocode** to implement the DeleteNode procedure.

```

PROCEDURE DeleteNode (NodeValue: STRING, ThisPointer : INTEGER,
                      PreviousPointer : INTEGER)

IF CountryList [ThisPointer].Value = NodeValue

THEN
    CountryList [ThisPointer].Value ← ""

    IF ListHead = ..... .

    THEN
        ListHead ← ......

    ELSE
        CountryList [PreviousPointer].Pointer ← CountryList [ThisPointer].Pointer
    ENDIF

    CountryList [LastNode].Pointer ← ......

    LastNode ← ThisPointer

    .....

ELSE
    IF CountryList [ThisPointer].Pointer <> -1

    THEN
        CALL DeleteNode (NodeValue, ..... ,
                        ThisPointer)

    ELSE
        OUTPUT "DOES NOT EXIST"
    ENDIF
ENDIF

ENDPROCEDURE

```

[5]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/41

Paper 4 Practical

October/November 2023

2 hours 30 minutes

You will need: Candidate source files (listed on page 2)
evidence.doc



INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the evidence document, **evidence.doc**

Make sure that your name, centre number and candidate number appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: **evidence_zz999_9999**

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

One source file is used to answer **Question 2**. The file is called **QueueData.txt**

- 1 This iterative pseudocode algorithm for the function `IterativeVowels()` takes a string as a parameter and counts the number of lower-case vowels in this string.
The vowels are the letters a, e, i, o and u.

```
FUNCTION IterativeVowels(Value : STRING) RETURNS INTEGER
    DECLARE Total : INTEGER
    DECLARE LengthString : INTEGER
    DECLARE FirstCharacter : CHAR
    Total ← 0
    LengthString ← LENGTH(Value)
    FOR X ← 0 TO LengthString - 1
        FirstCharacter ← MID(Value, 0, 1)
        IF FirstCharacter = 'a' OR FirstCharacter = 'e' OR
            FirstCharacter = 'i' OR FirstCharacter = 'o' OR
            FirstCharacter = 'u' THEN
            Total ← Total + 1
        ENDIF
        Value ← MID(Value, 1, LENGTH(Value)-1)
    NEXT X
    RETURN Total
ENDFUNCTION
```

The pseudocode function `MID(X, Y, Z)` returns `Z` number of characters from string `X`, starting at the character in position `Y`. The first character in a string is in position 0, for example:

`MID("computer", 0, 3)` returns "com"

The pseudocode function `LENGTH(X)` returns the number of characters in the string `X`, for example:

`LENGTH("computer")` returns 8

- (a) (i) Write program code for the function `IterativeVowels()`.

Save your program as **Question1_N23**.

Copy and paste the program code into part **1(a)(i)** in the evidence document.

[5]

- (ii) Write program code to call the function `IterativeVowels()` with the parameter "house" from the main program.

Output the return value.

Save your program.

Copy and paste the program code into part **1(a)(ii)** in the evidence document.

[2]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **1(a)(iii)** in the evidence document.

[1]

- (b) (i) Rewrite the function `IterativeVowels()` as a recursive function with the identifier `RecursiveVowels()`.

Save your program.

Copy and paste the program code into part **1(b)(i)** in the evidence document.

[6]

- (ii) Write program code to call the function `RecursiveVowels()` with the parameter "imagine" from the main program.

Output the return value.

Save your program.

Copy and paste the program code into part **1(b)(ii)** in the evidence document.

[1]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **1(b)(iii)** in the evidence document.

[1]

2 A linear queue is implemented using the 1D array, Queue. The index of the first element in the array is 0.

(a) (i) Write program code to declare:

- Queue — a global array with space to store 50 IDs of type string
- HeadPointer — a global variable to point to the first element in the queue, initialised to -1
- TailPointer — a global variable to point to the next available space in the queue, initialised to 0.

Save your program as **Question2_N23**.

Copy and paste the program code into part **2(a)(i)** in the evidence document.

[2]

(ii) The procedure Enqueue () takes a string parameter.

If the queue is full, the procedure outputs a suitable message. If the queue is not full, the procedure inserts the parameter into the queue and updates the relevant pointer(s).

Write program code for Enqueue () .

Save your program.

Copy and paste the program code into part **2(a)(ii)** in the evidence document.

[4]

(iii) The function Dequeue () checks if the queue is empty.

If the queue is empty, the function outputs a suitable message and returns the string "Empty".

If the queue is not empty, the function returns the first element in the queue and updates the relevant pointer(s).

Write program code for Dequeue () .

Save your program.

Copy and paste the program code into part **2(a)(iii)** in the evidence document.

[4]

- (b) A shop sells computer games. Each game has a unique identifier (ID) of string data type.

The text file `QueueData.txt` contains a list of game IDs.

The procedure `ReadData()` reads the data from the text file and inserts each item of data into the array `Queue`.

Write program code for the procedure `ReadData()`.

Save your program.

Copy and paste the program code into part 2(b) in the evidence document.

[6]

- (c) Some game IDs appear in the text file more than once.

The program needs to total the number of times each game ID appears in the text file.

The record structure `RecordData` has the following fields:

- `ID` — a string to store the game ID
- `Total` — an integer to store the total number of times that game ID appears in the text file.

- (i) Write program code to declare the record structure `RecordData`.

If you are writing in Python, include attribute declarations as comments.

Save your program.

Copy and paste the program code into part 2(c)(i) in the evidence document.

[2]

- (ii) The global 1D array `Records` stores up to 50 items of type `RecordData`.

The global variable `NumberRecords` stores the number of records currently in the array `Records` and is initialised to 0.

Write program code to declare `Records` and `NumberRecords`.

If you are writing in Python, include attribute declarations as comments.

Save your program.

Copy and paste the program code into part 2(c)(ii) in the evidence document.

[2]

(iii) The pseudocode algorithm for the procedure TotalData():

- uses Dequeue() to remove an ID from the queue
- checks whether a RecordData with the returned ID exists in Records
- increments the total for that ID in the record if the ID already exists
- creates a new record and stores it in Records if the ID does not exist.

```

PROCEDURE TotalData()

DECLARE DataAccessed : STRING
DECLARE Flag : BOOLEAN
DataAccessed ← Dequeue()
Flag ← FALSE
IF NumberRecords = 0 THEN
    Records[NumberRecords].ID ← DataAccessed
    Records[NumberRecords].Total ← 1
    Flag ← TRUE
    NumberRecords ← NumberRecords + 1
ELSE
    FOR X ← 0 TO NumberRecords - 1
        IF Records[X].ID = DataAccessed THEN
            Records[X].Total ← Records[X].Total + 1
            Flag ← TRUE
        ENDIF
    NEXT X
ENDIF
IF Flag = FALSE THEN
    Records[NumberRecords].ID ← DataAccessed
    Records[NumberRecords].Total ← 1
    NumberRecords ← NumberRecords + 1
ENDIF
ENDPROCEDURE

```

Write program code for the procedure TotalData().

Save your program.

Copy and paste the program code into part 2(c)(iii) in the evidence document.

[5]

- (d) The procedure OutputRecords () outputs the ID and total of each record in Records in the format:

ID 1234 Total 4

Write program code for OutputRecords () .

Save your program.

Copy and paste the program code into part 2(d) in the evidence document.

[1]

- (e) The main program needs to:

- call ReadData ()
- call TotalData () for each element in the queue
- call OutputRecords () .

(i) Write program code for the main program.

Save your program.

Copy and paste the program code into part 2(e)(i) in the evidence document.

[2]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part 2(e)(ii) in the evidence document.

[1]

- 3 A computer game is written using object-oriented programming.

The game has multiple characters that can move around the screen.

The class `Character` stores data about the characters. Each character has a name, a current X (horizontal) position and a current Y (vertical) position.

Character	
Name : STRING	stores the name of the character as a string
XPosition : INTEGER	stores the X position as an integer
YPosition : INTEGER	stores the Y position as an integer
Constructor()	initialises Name, XPosition and YPosition to its parameter values
GetXPosition()	returns the X position
GetYPosition()	returns the Y position
SetXPosition()	adds the parameter to the X position validates that the new X position is between 0 and 10 000 inclusive
SetYPosition()	adds the parameter to the Y position validates that the new Y position is between 0 and 10 000 inclusive
Move()	takes a direction as a parameter and calls either SetXPosition or SetYPosition with an integer value

- (a) (i) Write program code to declare the class `Character` and its constructor.

Do **not** declare the other methods.

Use your programming language's appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_N23**.

Copy and paste the program code into part 3(a)(i) in the evidence document.

[4]

- (ii) The get methods `GetXPosition()` and `GetYPosition()` each return the relevant attribute.

Write program code for the get methods.

Save your program.

Copy and paste the program code into part 3(a)(ii) in the evidence document.

[3]

- (iii) The set methods `SetXPosition()` and `SetYPosition()` each take a value as a parameter and add this to the current X or Y position.

If the new value exceeds 10 000, it is limited to 10 000.

If the new value is below 0, it is limited to 0.

Write program code for the set methods.

Save your program.

Copy and paste the program code into part 3(a)(iii) in the evidence document.

[4]

- (iv) The method `Move()` takes a string parameter: "up", "down", "left" or "right".

The table shows the change each direction will make to the X or Y position.

Use the appropriate method to change the position value.

Direction	Value change
up	Y position + 10
down	Y position - 10
left	X position - 10
right	X position + 10

Write program code for `Move()`.

Save your program.

Copy and paste the program code into part 3(a)(iv) in the evidence document.

[4]

- (b) Write program code to declare a new instance of `Character` with the identifier `Jack`.

The starting X position is 50 and the starting Y position is 50, the character's name is Jack.

Save your program.

Copy and paste the program code into part 3(b) in the evidence document.

[2]

- (c) The class `BikeCharacter` inherits from the class `Character`.

BikeCharacter	
<code>Constructor()</code>	takes <code>Name</code> , <code>XPosition</code> and <code>YPosition</code> as parameters calls its parent class constructor with the appropriate values
<code>Move()</code>	overrides the method <code>Move()</code> from the parent class by changing either the X position or the Y position by 20 instead of 10

- (i) Write program code to declare the class `BikeCharacter` and its constructor.

Do **not** declare the other method.

Use your programming language's appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into part 3(c)(i) in the evidence document.

[3]

- (ii) The method `Move()` overrides the method from the parent class.

The table shows the change each direction will make to the X or Y position.

Direction	Value change
up	Y position + 20
down	Y position - 20
left	X position - 20
right	X position + 20

Write program code for `Move()`.

Save your program.

Copy and paste the program code into part 3(c)(ii) in the evidence document.

[2]

- (d) Write program code to declare a new instance of `BikeCharacter` with the identifier `Karla`.

The starting X position is 100, the starting Y position is 50 and the character's name is Karla.

Save your program.

Copy and paste the program code into part 3(d) in the evidence document.

[1]

(e) (i) Write program code to:

- take as input which of the two characters the user would like to move
- take as input the direction the user would like the character to move
- call the appropriate method to move the character
- output the character's new X and Y position in an appropriate format, for example:
"Karla's new position is X = 100 Y = 200"

All inputs require appropriate prompts and must be validated.

Save your program.

Copy and paste the program code into part 3(e)(i) in the evidence document.

[5]

(ii) Test your program twice with the following inputs.

Test 1: jack right

Test 2: karla down

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part 3(e)(ii) in the evidence document.

[2]

BLANK PAGE

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.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

May/June 2016

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **19** printed pages and **1** blank page.

- 1 A linked list abstract data type (ADT) is to be used to store and organise surnames.

This will be implemented with a 1D array and a start pointer. Elements of the array consist of a user-defined type. The user-defined type consists of a data value and a link pointer.

Identifier	Data type	Description
LinkedList	RECORD	User-defined type
Surname	STRING	Surname string
Ptr	INTEGER	Link pointers for the linked list

- (a) (i) Write **pseudocode** to declare the type `LinkedList`.

.....
.....
.....
.....

[3]

- (ii) The 1D array is implemented with an array `SurnameList` of type `LinkedList`.

Write the **pseudocode** declaration statement for `SurnameList`. The lower and upper bounds of the array are 1 and 5000 respectively.

..... [2]

- (b) The following surnames are organised as a linked list with a start pointer `StartPtr`.

`StartPtr: 3`

	1	2	3	4	5	6	5000
Surname	Liu	Yang	Chan	Wu	Zhao	Huang	...	
Ptr	4	5	6	2	0	1	...	

State the value of the following:

- (i) `SurnameList[4].Surname` [1]

- (ii) `SurnameList[StartPtr].Ptr` [1]

- (c) Pseudocode is to be written to search the linked list for a surname input by the user.

Identifier	Data type	Description
ThisSurname	STRING	The surname to search for
Current	INTEGER	Index to array SurnameList
StartPtr	INTEGER	Index to array SurnameList. Points to the element at the start of the linked list

- (i) Study the pseudocode in part (c)(ii).

Complete the table above by adding the missing identifier details.

[2]

- (ii) Complete the pseudocode.

```

01 Current ← .....
02 IF Current = 0
03   THEN
04     OUTPUT .....
05 ELSE
06   IsFound ← .....
07   INPUT ThisSurname
08 REPEAT
09   IF ..... = ThisSurname
10     THEN
11       IsFound ← TRUE
12       OUTPUT "Surname found at position ", Current
13     ELSE
14       // move to the next list item
15       .....
16   ENDIF
17 UNTIL IsFound = TRUE OR .....
18 IF IsFound = FALSE
19   THEN
20   OUTPUT "Not Found"
21 ENDIF
22 ENDIF

```

[6]

- 2 (a) (i) State what is meant by a recursively defined procedure.

.....
.....

[1]

- (ii) Write the line number from the pseudocode shown in part (b) that shows the procedure X is recursive.

[1]

- (b) The recursive procedure X is defined as follows:

```

01 PROCEDURE X(Index, Item)
02     IF MyList[Index] > 0
03         THEN
04             IF MyList[Index] >= Item
05                 THEN
06                     MyList[Index] ← MyList[Index + 1]
07                 ENDIF
08             CALL X(Index + 1, Item)
09         ENDIF
10     ENDPROCEDURE

```

An array MyList is used to store a sorted data set of non-zero integers. Unused cells contain zero.

	1	2	3	4	5	6	7	8	9	10
MyList	3	5	8	9	13	16	27	0	0	0

- (i) Complete the trace table for the dry-run of the pseudocode for the procedure
CALL X(1, 9).

		MyList									
Index	Item	1	2	3	4	5	6	7	8	9	10
1	9	3	5	8	9	13	16	27	0	0	0

[4]

- (ii) State the purpose of procedure X when used with the array MyList.

.....
.....

[1]

- 3 A car hire company hires cars to customers. Each time a car is hired, this is treated as a transaction.

For each transaction, the following data are stored.

For the customer:

- customer name
- ID number

For the hire:

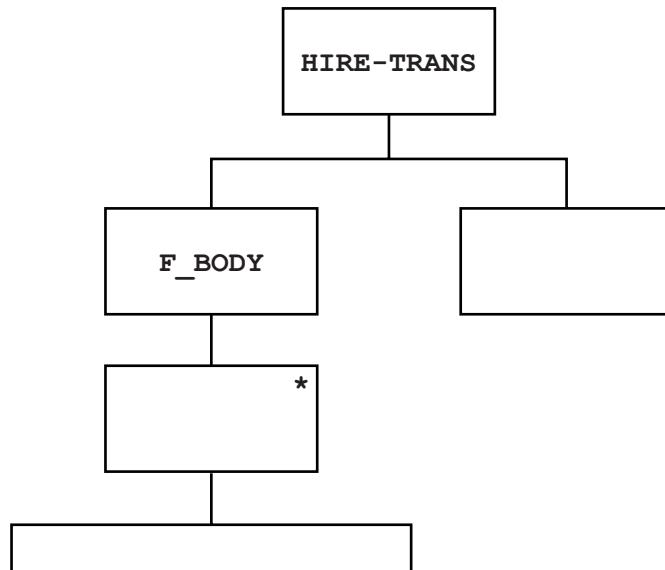
- car registration
- hire start date
- number of days hired

The transaction data are stored in a text file HIRE-TRANS. The file is made up of a file body, F_BODY, and a file trailer, F_TRAILER.

F_BODY has one transaction, TRANS, on each line.

- (a) The first step in Jackson Structured Programming (JSP) design is to produce a JSP data structure diagram.

Complete the following JSP data structure diagram.



- (b) The computer system will produce many printed reports.

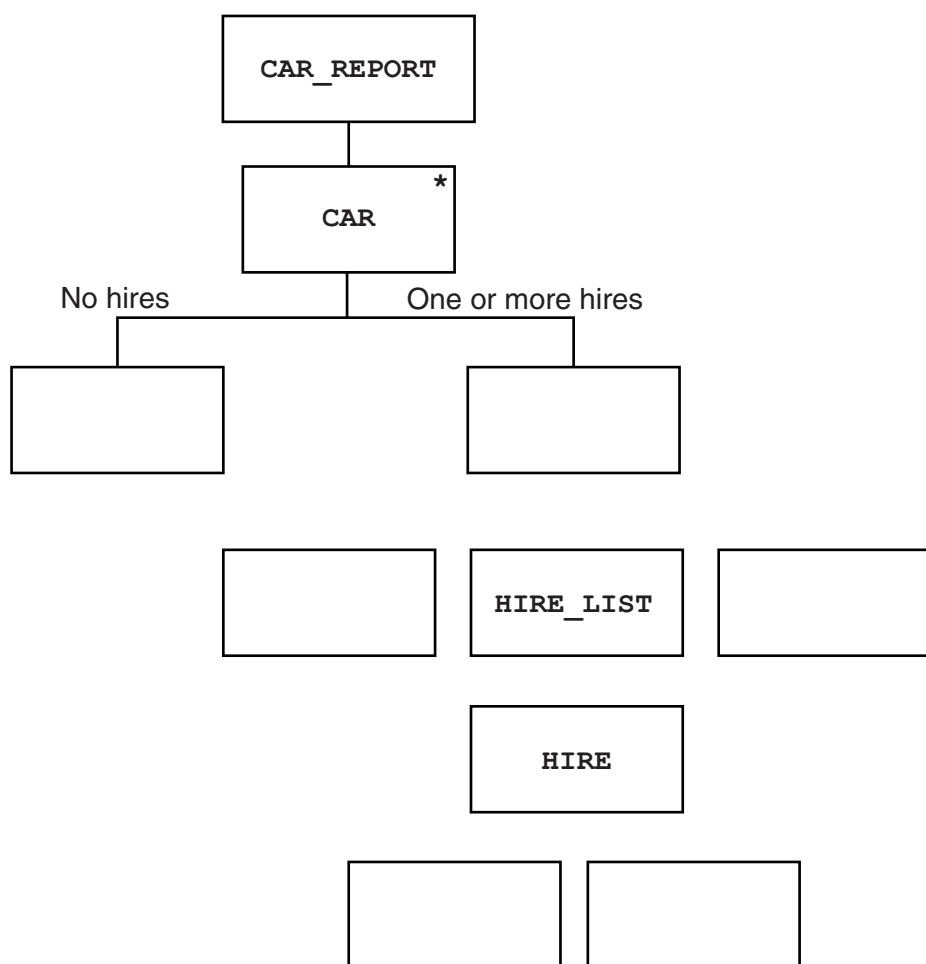
One report is CAR_REPORT. This displays all hire data for all cars.

For each car, the following data are displayed:

- the car data
- a list of all the hires
- the total number of hires

A car with zero hires is not included on the report.

Complete the following CAR_REPORT JSP data structure diagram.



[5]

- 4 When a car reaches a certain age, a safety assessment has to be carried out. A car's brakes and tyres must be tested. The tyre test result and the brakes test result for each car are recorded. If the car passes the assessment, a safety certificate is issued.

Cars have a unique three-character registration.

The following knowledge base is used:

```

01 car(a05).
02 car(h04).
03 car(a03).
04 car(h07).
05 car(a23).
06 car(p05).
07 car(b04).
08 carRegYear(a05, 2015).
09 carRegYear(h04, 2013).
10 carRegYear(a03, 2008).
11 carRegYear(h07, 2011).
12 carRegYear(a23, 2008).
13 carRegYear(p05, 2014).
14 carRegYear(b04, 2014).
15 testBrakes(h07, pass).
16 testTyres(h07, fail).
17 testBrakes(a03, fail).
18 testTyres(a03, fail).
19 testBrakes(a23, pass).
20 testTyres(a23, pass).
21 carAssessmentDue if carRegYear(Car, RegYear)
                           and RegYear <= DeadlineYear.
22 issueCertificate(Car) if testTyres(Car, Result) and
                           testBrakes(Car, Result) and Result = pass.

```

- (a) (i) DeadlineYear is assigned value 2011.

Identify the car registrations for cars which are due to be tested.

..... [1]

- (ii) State how clause 22 determines whether or not a safety certificate will be issued.

..... [1]

- (b)** If a car fails one of the two tests, a retest is allowed.

Write a new rule for this.

retestAllowed(.....) if

.....

..... [3]

- (c)** Logic programming uses a data structure called a list.

A new fact is added to the knowledge base.

23 carList = [a03, p05, b04, h04, h07, a23].

The following notation and operators are to be used with a list:

[X | Y] denotes a list with:

- X the first list element
- Y the list consisting of the remaining list elements

[] denotes an empty list

- (i)** The list [a07, p03] is denoted by [A | B]

State the value of A and B.

A =

B = [2]

- (ii)** The lists [c03, d02, n05 | C] and [c03, d02, n05, p05, m04] are identical.

State the value of C.

C = [1]

- (iii)** The list [a06, a02] is denoted by [D, E | F]

State the value of F.

F = [1]

- (d) The predicate `conCatCompare` is defined as a rule and returns TRUE or FALSE as follows:

`conCatCompare(X, Y, Z)`

Concatenates the lists X and Y and compares the new list with list Z.

If equal, the clause evaluates to TRUE, otherwise FALSE.

Consider the clause:

`conCatCompare(X, Y, [a7,b6,c4])`

If:

- the clause evaluates to TRUE
- and Y represents the list [a7, b6, c4]

State the value of X.

X = [1]

- 5 (a) A program calculates the exam grade awarded from a mark input by the user. The code is written as a function CalculateGrade.

The function:

- has a single parameter **Mark** of **INTEGER** data type
- returns the grade awarded **Grade** of **STRING** data type

The logic for calculating the grade is as follows:

Mark	Grade
Under 40	FAIL
40 and over and under 55	PASS
55 and over and under 70	MERIT
70 and over	DISTINCTION

The programmer designs the following table for test data:

Mark	Description	Expected result (Grade)
	Normal	
	Abnormal	
	Extreme/Boundary	

- (i) Complete the table above. [3]
- (ii) State why this table design is suitable for black box testing.

..... [1]

(b) When designing and writing program code, explain what is meant by:

- an exception
- exception handling

.....
.....
.....
.....
.....
.....

[3]

(c) A program is to be written to read a list of exam marks from an existing text file into a 1D array.

Each line of the file stores the mark for one student.

State **three** exceptions that a programmer should anticipate for this program.

1

.....

2

.....

3

.....

[3]

- (d) The following pseudocode is to read two numbers:

```

01  DECLARE Num1      : INTEGER
02  DECLARE Num2      : INTEGER
03  DECLARE Answer : INTEGER
04  TRY
05      OUTPUT "First number..."
06      INPUT Num1
07      OUTPUT "Second number..."
08      INPUT Num2
09      Answer ← Num1 / (Num2 - 6)
10      OUTPUT Answer
11  EXCEPT ThisException : EXCEPTION
12      OUTPUT ThisException.Message
13  FINALLY
14      // remainder of the program follows
...
29
30 ENDTRY

```



The programmer writes the corresponding program code.

A user inputs the number 53 followed by 6. The following output is produced:

```

First number...53
Second number...6
Arithmetic operation resulted in an overflow

```

- (i) State the pseudocode line number which causes the exception to be raised.

.....

[1]

- (ii) Explain the purpose of the pseudocode on lines 11 and 12.

.....

[3]

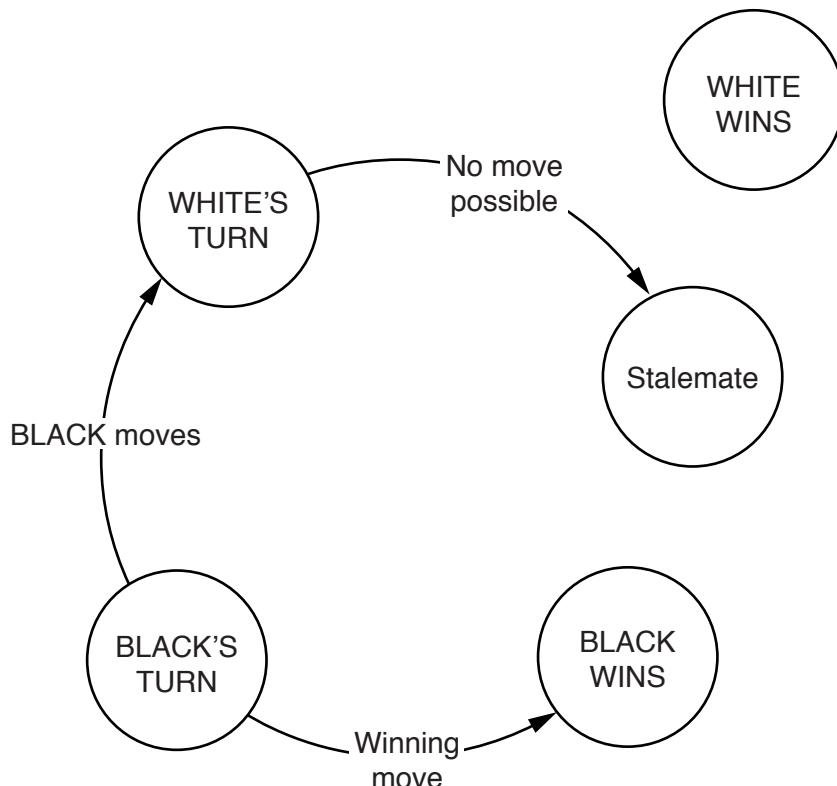
- 6 In a board game, one player has white pieces and the other player has black pieces. Players take alternate turns to move one of their pieces. White always makes the first move.

The game ends if:

- a player is unable to make a move when it is their turn. In this case, there is no winner. This is called 'stalemate'.
- a player wins the game as a result of their last move and is called a 'winner'.

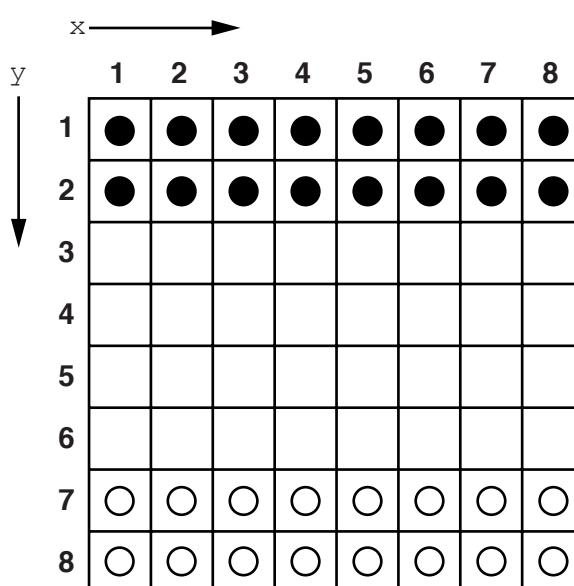
- (a) A state-transition diagram is drawn to clarify how the game is played.

Complete the following state-transition diagram.



[4]

- (b) The layout of the board at the start of the game is shown below:



The programmer decides to use a 2D array to represent the board. The index numbering to be used is as shown.

Each square on the board is either occupied by one piece only, or is empty.

The data stored in the array indicate whether or not that square is occupied, and if so, with a black piece or a white piece.

- (i) Write **program code** to initialise the contents of the array to represent the board at the start of the game. Use characters as follows for each square:

- 'B' represents a black piece 
 - 'W' represents a white piece 
 - 'E' represents an empty square

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language

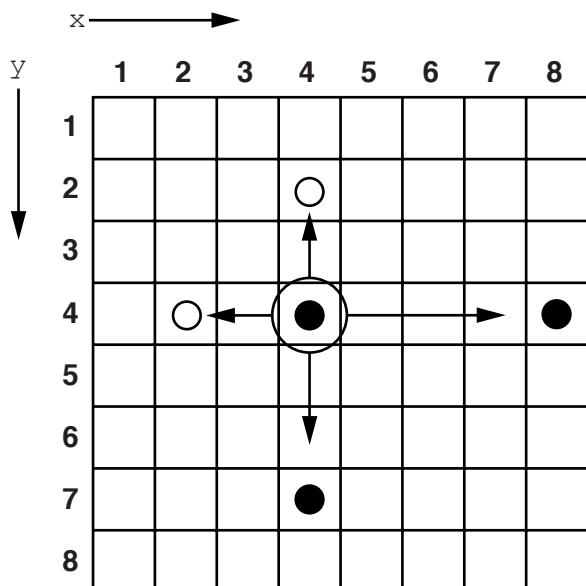
- (ii) When a piece is to be moved, a procedure will calculate and output the possible destination squares for the moving piece.

A piece can move one or more squares, in the x or y direction, from its current position.

This will be a move:

- either to an empty square, with no occupied squares on the way
- or to a square containing a piece belonging to another player, with no occupied squares on the way. The other player's piece is then removed.

For example, for the circled black piece there are nine possible destination squares. Each of the two destination squares contains a white piece which would be removed.



The program requires a procedure `ValidMoves`.

It needs three parameters:

- `PieceColour` – colour of the moving piece
- `xCurrent` – current x position
- `yCurrent` – current y position

The procedure will calculate all possible destination squares **in the x direction only**.

Example output for the circled black piece is:

```

Possible moves are:
Moving LEFT
3 4
2 4 REMOVE piece
Moving RIGHT
5 4
6 4
7 4

```

Write **program code** for procedure `ValidMoves` with the following procedure header:

```

PROCEDURE ValidMoves(PieceColour : CHAR, xCurrent : INTEGER,
                      yCurrent : INTEGER).

```

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language

. [5]

- (c) The problem is well suited to an object-oriented design followed by object-oriented programming.

- (i) Describe how classes and objects could be used in this problem.

.....
.....
.....
.....
.....

[2]

- (ii) For a class you identified in part(c)(i), state **two** properties and **two** methods.

Class

Properties

1
2

Methods

1,
2

[2]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

--	--	--	--	--

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

October/November 2016

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

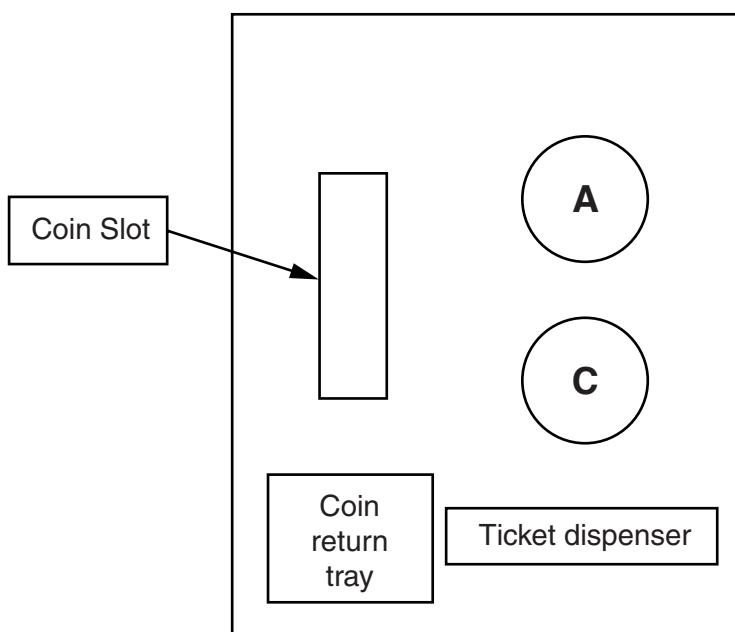
The maximum number of marks is 75.

This document consists of 17 printed pages and 3 blank pages.

- 1 The ticket machine in the following diagram accepts the following coins: 10, 20, 50 and 100 cents.

The ticket machine has:

- a slot to insert coins
- a tray to return coins
- a ticket dispenser
- two buttons:
 - button **A** (Accept)
 - button **C** (Cancel)



When the user has inserted as many coins as required, they press button **A** to print the ticket.

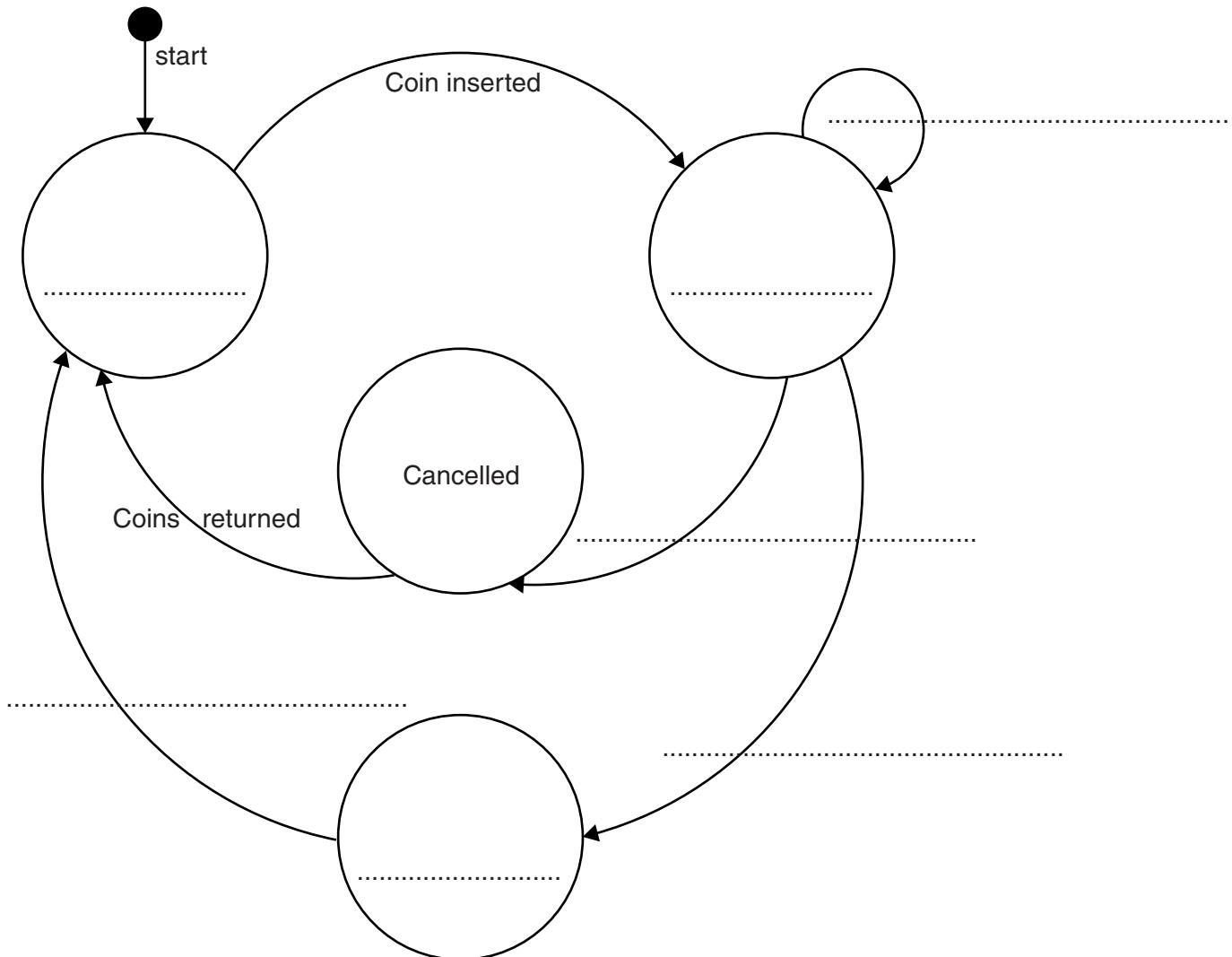
To cancel the transaction, the user can press button **C**. This makes the machine return the coins.

Invalid coins have no effect.

The following state transition table shows the transition from one state to another of the ticket machine:

Current state	Event	Next state
Idle	Coin inserted	Counting
Counting	Coin inserted	Counting
Counting	Button C pressed	Cancelled
Cancelled	Coins returned	Idle
Counting	Button A pressed	Accepted
Accepted	Ticket printed	Idle

- (a) Complete the state-transition diagram.



[7]

- (b) A company wants to simulate the use of a ticket machine. It will do this with object-oriented programming (OOP).

The following diagram shows the design for the class `TicketMachine`. This includes its attributes and methods.

TicketMachine	
Amount : INTEGER	// total value of coins inserted in cents
State : STRING	// "Idle", "Counting", "Cancelled" // or "Accepted"
Create()	// method to create and initialise an object // if using Python use <code>__init__</code>
SetState()	// set state to parameter value // and output new state
StateChange()	// insert coin or press button, // then take appropriate action
CoinInserted()	// parameter is a string // change parameter to integer // and add coin value to Amount
ReturnCoins()	// output Amount, then set Amount to zero
PrintTicket()	// print ticket, then set Amount to zero

Write **program code** for the following methods.

Programming language

(i) `Create()`

.....
.....
.....
.....
.....
.....

[3]

(ii) `SetState()`

.....
.....
.....
.....

[2]

- (iii) ReturnCoins()

[2]

[2]

- (iv) Each coin inserted must be one of the following: 10, 20, 50 or 100 cents.

Write **program code** for a function ValidCoin(s : STRING) that returns:

- TRUE if the input string is one of "10", "20", "50" or "100"
 - FALSE otherwise

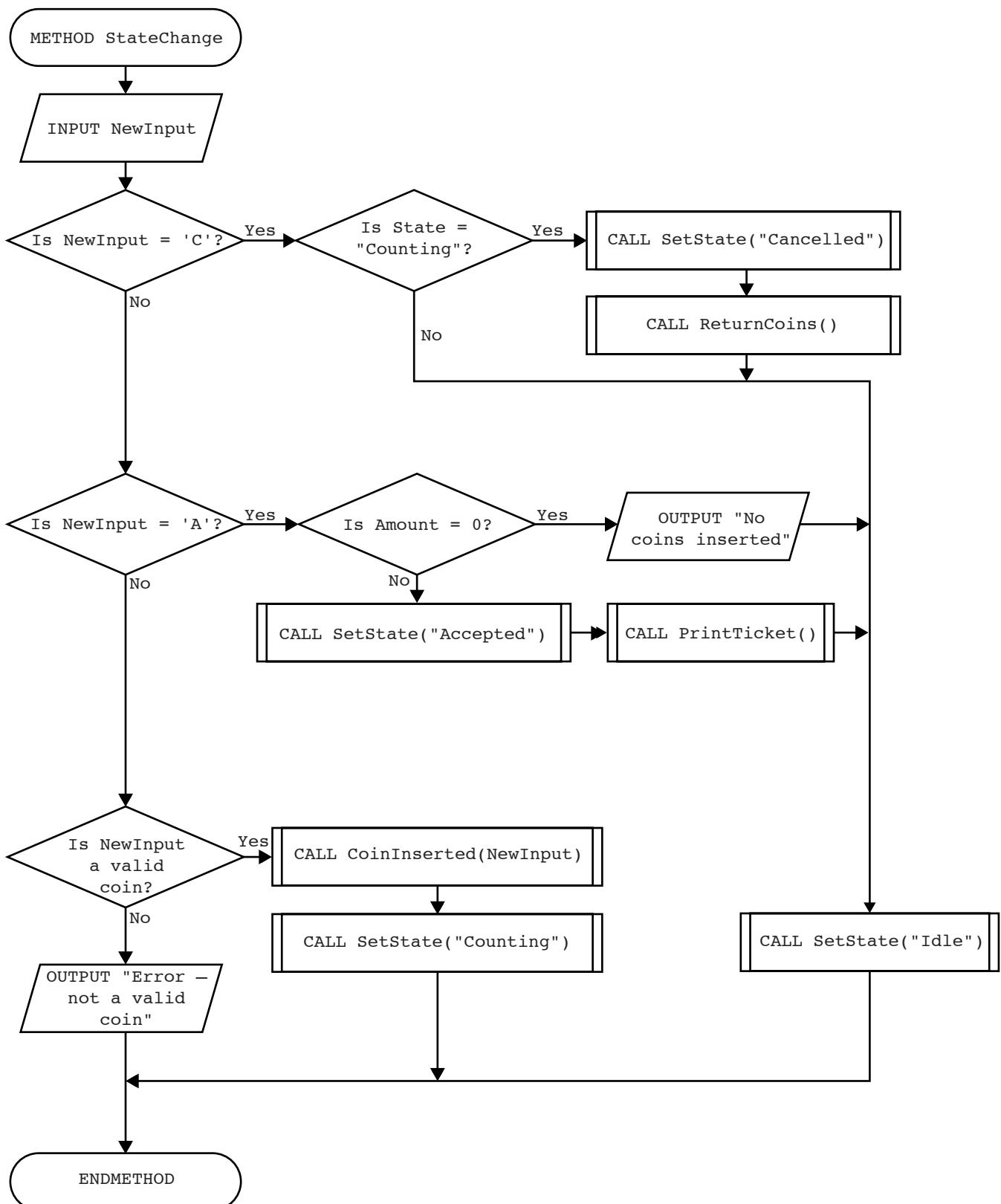
- [3]

- (v) Write **program code** for the method `CoinInserted()`

[2]

-[2]

- (vi) Convert the flowchart to **program code** for the method `StateChange()`.
 Use the attributes and methods in the original class definition and the `ValidCoin()` function from **part (iv)**.



Programming language

.[12]

- (vii) The company needs to write a program to simulate a parking meter. The program will create an object with identifier ParkingMeter, which is an instance of the class TicketMachine.

The main program design is:

```
instantiate ParkingMeter (create and initialise ParkingMeter)
loop forever (continually use ParkingMeter)
    call StateChange() method
end loop
```

Write **program code** for the main program.

Programming language

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

[4]

- (c) It is possible to declare attributes and methods as either public or private.

A programmer has modified the class design for TicketMachine as follows.

TicketMachine
PRIVATE
Amount : INTEGER
State : STRING
PUBLIC
Create()
StateChange()
PRIVATE
SetState()
CoinInserted()
ReturnCoins()
PrintTicket()

- (i) Describe the effects of declaring the TicketMachine attributes as private.

.....

 [2]

- (ii) Describe the effects of declaring two methods of the class as public and the other four as private.

.....

 [2]

- 2** Commercial software usually undergoes alpha testing and beta testing.

Distinguish between the two types of testing by stating:

- who does the testing
- when the testing occurs
- the specific purpose of each type of testing

(i) Alpha testing

Who

When

Purpose

..... [3]

(ii) Beta testing

Who

When

Purpose

..... [3]

- 3 (a)** The numerical difference between the ASCII code of an upper case letter and the ASCII code of its lower case equivalent is 32 denary (32_{10}).

For example, 'F' has ASCII code 70 and 'f' has ASCII code 102.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0

The bit patterns differ only at bit number 5. This bit is 1 if the letter is lower case and 0 if the letter is upper case.

- (i) A program needs a mask to ensure that a letter is in **upper case**.

Write the binary pattern of the mask in the space provided in the table below.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0
Mask								

Give the bit-wise operation that needs to be performed using the mask and the ASCII code.

..... [2]

- (ii) A program needs a mask to ensure that a letter is in **lower case**.

Write the binary pattern of the mask in the space provided in the table below.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0
Mask								

Give the bit-wise operation that needs to be performed using the mask and the ASCII code.

..... [2]

The following table shows part of the instruction set for a processor which has one general purpose register, the Accumulator (ACC), and an index register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the given address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n into IX.
STO	<address>	Store the contents of ACC at the given address.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

A programmer is writing a program that will output the first character of a string in upper case and the remaining characters of the string in lower case.

The program will use locations from address WORD onwards to store the characters in the string. The location with address LENGTH stores the number of characters that make up the string.

The programmer has started to write the program in the following table. The comment column contains descriptions for the missing program instructions.

- (b) Complete the program using op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// initialise index register to zero
			// get first character of WORD
			// ensure it is in upper case using MASK1
			// output character to screen
			// increment index register
			// load 1 into ACC
			// store in COUNT
LOOP:			// load next character from indexed address WORD
			// make lower case using MASK2
			// output character to screen
			// increment COUNT starts here
			// is COUNT = LENGTH ?
			// if FALSE, jump to LOOP
			// end of program
COUNT:			
MASK1:			// bit pattern for upper case
MASK2:			// bit pattern for lower case
LENGTH:	4		
WORD:		B01100110	// ASCII code in binary for 'f'
		B01110010	// ASCII code in binary for 'r'
		B01000101	// ASCII code in binary for 'E'
		B01000100	// ASCII code in binary for 'D'

[12]

Question 4 begins on page 15.

- 4 Circle the programming language that you have studied:

Visual Basic (console mode) Python Pascal Delphi (console mode)

- (a) (i) Name the programming environment you have used when typing in program code.

.....
.....

List **three** features of the editor that helped you to write program code.

1

.....

2

.....

3

..... [3]

- (ii) Explain when and how your programming environment reports a syntax error.

When

.....
.....

How

.....
.....

[2]

(iii) The table shows a module definition for BubbleSort in three programming languages.

Study **one** of the examples. Indicate your choice by circling A, B or C:

A B C

	A) Python
01	def BubbleSort(SList, Max): 02 NoMoreSwaps = False 03 while NoMoreSwaps == False: 04 NoMoreSwaps = True 05 for i in (Max - 1): 06 if SList[i] > SList[i + 1]: 07 NoMoreSwaps = True 08 Temp = SList[i] 09 SList[i] = SList[i + 1] 10 SList[i + 1] = Temp
	B) Pascal/Delphi
01	PROCEDURE BubbleSort(VAR SList : ARRAY OF INTEGER; Max : INTEGER); 02 VAR NoMoreSwaps : BOOLEAN; i, Temp : INTEGER; 03 BEGIN 04 REPEAT 05 NoMoreSwaps := TRUE; 06 FOR i := 1 TO (Max - 1) 07 IF SList[i] > SList[i + 1] 08 THEN 09 BEGIN 10 NoMoreSwaps := TRUE; 11 Temp := SList[i]; 12 SList[i] := SList[i + 1]; 13 SList[i + 1] := Temp; 14 END; 15 UNTIL NoMoreSwaps; 16 END;
	C) Visual Basic
01	Sub BubbleSort(ByRef SList() As Integer, ByVal Max As Integer) 02 Dim NoMoreSwaps As Boolean, i, Temp As Integer 03 Do 04 NoMoreSwaps = True 05 For i : 0 To (Max - 1) 06 If SList(i) > SList(i + 1) Then 07 NoMoreSwaps = True 08 Temp = SList(i) 09 SList(i) = SList(i + 1) 10 SList(i + 1) = Temp 11 End If 12 Next 13 Loop Until (NoMoreSwaps = True) 14 End Sub

The programming environment reported a syntax error in the `BubbleSort` code.

State the line number

Write the correct code for this line.

..... [2]

- (b) (i) State whether programs written in your programming language are compiled or interpreted.

.....

[1]

- (ii) A programmer corrects the syntax error and tests the function. It does not perform as expected. The items are not fully in order.

State the type of error

Write the line number where the error occurs.

.....

Write the correct code for this line.

..... [2]

- (iii) State the programming environment you have used when debugging program code.

.....

Name **two** debugging features and describe how they are used.

1

.....

.....

.....

2

.....

.....

..... [4]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

October/November 2016

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

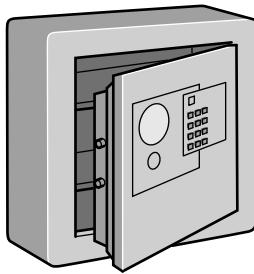
At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

- 1 A user can lock a safety deposit box by inputting a 4-digit code. The user can unlock the box with the same 4-digit code.



There is a keypad on the door of the safety deposit box. The following diagram shows the keys on the keypad.

1	2	3
4	5	6
7	8	9
R	0	Enter

Initially, the safety deposit box door is open and the user has not set a code.

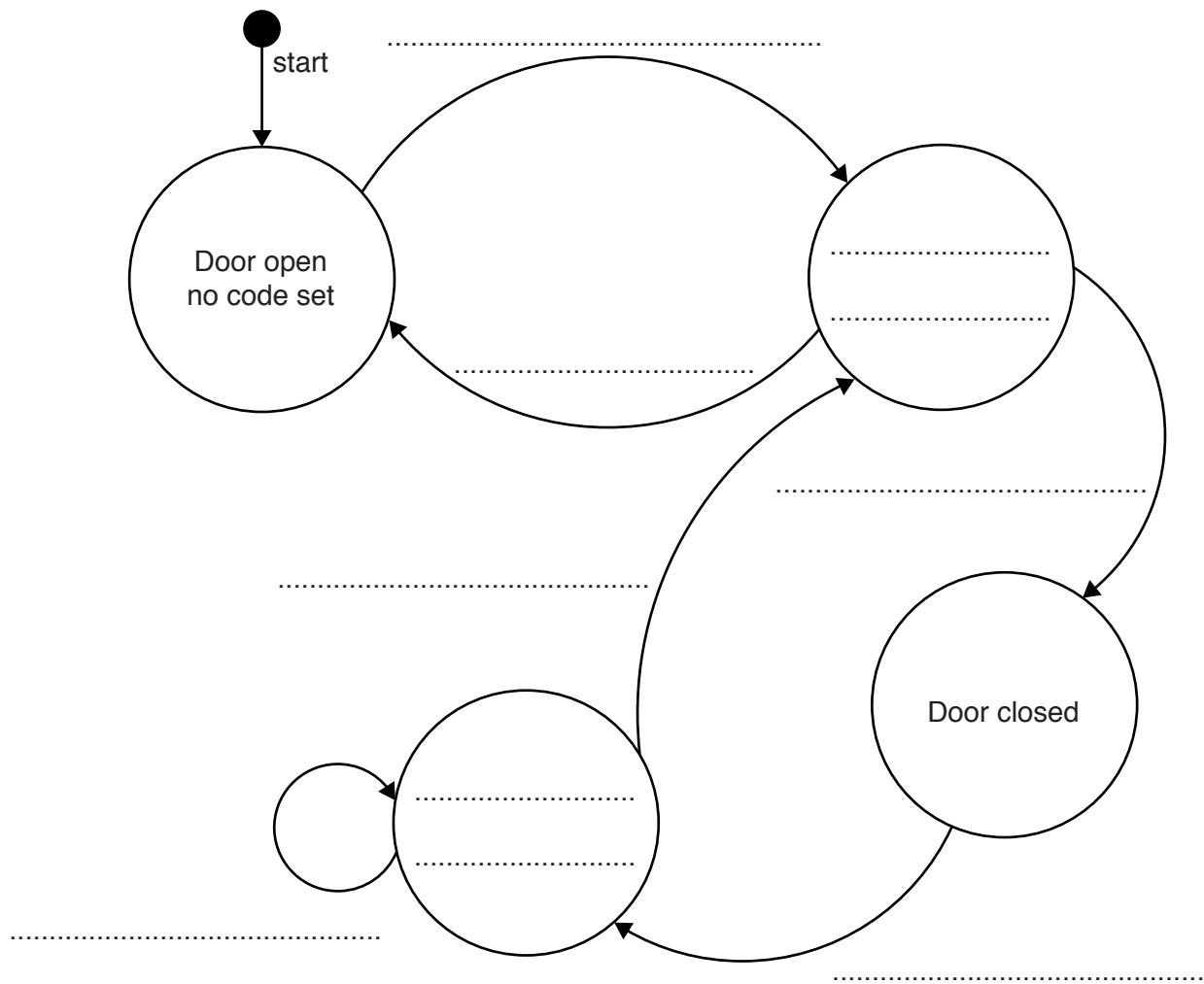
The operation of the safety deposit box is as follows:

- A) To set a new code the door must be open. The user chooses a 4-digit code and sets it by pressing the numerical keys on the keypad, followed by the Enter key. Until the user clears this code, it remains the same. (See point E below)
- B) The user can only close the door if the user has set a code.
- C) To lock the door, the user closes the door, enters the set code and presses the Enter key.
- D) To unlock the door, the user enters the set code. The door then opens automatically.
- E) The user clears the code by opening the door and pressing the R key, followed by the Enter key. The user can then set a new code. (See point A above)

The following state transition table shows the transition from one state to another of the safety deposit box:

Current state	Event	Next state
Door open, no code set	4-digit code entered	Door open, code set
Door open, code set	R entered	Door open, no code set
Door open, code set	Close door	Door closed
Door closed	Set code entered	Door locked
Door locked	Set code entered	Door open, code set
Door locked	R entered	Door locked

(a) Complete the state-transition diagram.



[7]

- (b) A company wants to simulate the use of a safety deposit box. It will do this with object-oriented programming (OOP).

The following diagram shows the design for the class `SafetyDepositBox`. This includes the properties and methods.

SafetyDepositBox	
Code	: STRING // 4 digits
State	: STRING // "Open-NoCode", "Open-CodeSet", "Closed" // or "Locked"
Create()	// method to create and initialise an object // if using Python use <code>__init__</code>
Reset()	// clears Code
SetState()	// set state to parameter value // and output new state
SetNewCode()	// sets Code to parameter value // output message and new code
StateChange()	// reads keypad and takes appropriate action

Write **program code** for the following methods.

Programming language

(i) Create()

.....
.....
.....
.....
.....

[3]

(ii) Reset()

.....
.....
.....

[2]

(iii) SetState()

[2]

[2]

(iv) SetNewCode()

[2]

[2]

(v) The user must enter a 4-digit code.

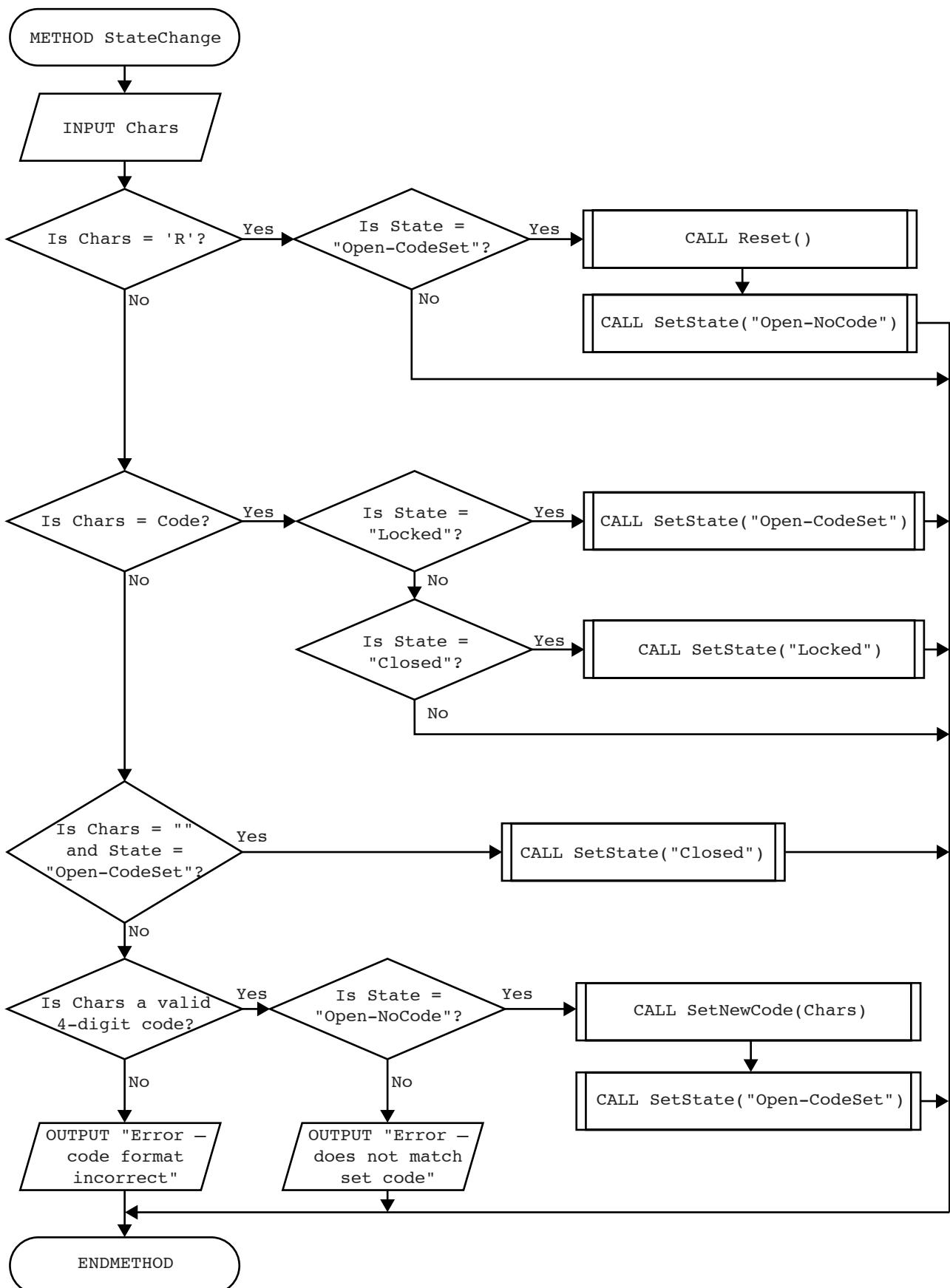
Write program code for a function Valid(s : STRING) that returns:

- TRUE if the input string s consists of exactly 4 digits
 - FALSE otherwise

Programming language

[4]

- (vi) Convert the flowchart to **program code** for the method `StateChange()`. Use the properties and methods in the original class definition and the `Valid()` function from part (v).



Programming language

[12]

- (vii) The company needs to write a program to simulate a safety deposit box. The program will create an object with identifier ThisSafe, which is an instance of the class SafetyDepositBox.

The main program design is:

```
instantiate ThisSafe (create and initialise ThisSafe)
loop forever (continually use ThisSafe)
    call StateChange() method
end loop
```

Write **program code** for the main program.

Programming language

[4]

- (c) It is possible to declare properties and methods as either public or private.

The programmer has modified the class design for `SafetyDepositBox` as follows:

SafetyDepositBox	
PRIVATE	
Code	: STRING
State	: STRING
PUBLIC	
Create()	
StateChange()	
PRIVATE	
Reset()	
SetState()	
SetNewCode()	

- (i) Describe the effects of declaring the `SafetyDepositBox` properties as private.

.....

 [2]

- (ii) Describe the effects of declaring two methods of the class as public and the other three as private.

.....

 [2]

2 Circle the programming language that you have studied:

Visual Basic (console mode) Python Pascal Delphi (console mode)

(a) (i) Name the programming environment you have used when typing in program code.

.....
.....

List **three** features of the editor that helped you to write program code.

1

.....

2

.....

3

..... [3]

(ii) Explain when and how your programming environment reports a syntax error.

When

.....

How

.....

[2]

Question 2 continues on page 12.

(iii) The table shows a module definition for BinarySearch in three programming languages.

Study one of the examples. Indicate your choice by circling A, B or C:

A **B** **C**

	A) Python
01	def BinarySearch(List, Low, High, SearchItem): Index = -1 while (Index == -1) AND (Low <= High): Middle = (High + Low) // 2 if List[Middle] == SearchItem: Index = Middle elif List[Middle] < SearchItem: Low = Middle + 1 else: High = Middle - 1 return(Middle)
	B) Pascal/Delphi
01	FUNCTION BinarySearch(VAR List : ARRAY OF INTEGER; Low, High, SearchItem : INTEGER) : INTEGER; 02 VAR Index, Middle : INTEGER; 03 BEGIN 04 Index := -1; 05 WHILE (Index = -1) & (Low <= High) DO 06 BEGIN 07 Middle := (High + Low) DIV 2; 08 IF List[Middle] = SearchItem 09 THEN Index := Middle 10 ELSE IF List[Middle] < SearchItem 11 THEN Low := Middle + 1 12 ELSE High := Middle - 1; 13 END; 14 Result := Middle; 15 END;
	C) Visual Basic
01	Function BinarySearch(ByRef List() As Integer, ByVal Low As Integer, ByVal High As Integer, ByVal SearchItem As Integer) As Integer 02 Dim Index, Middle As Integer 03 Index = -1 04 Do While (Index = -1) & (Low <= High) 05 Middle = (High + Low) \ 2 06 If List(Middle) = SearchItem Then 07 Index = Middle 08 ElseIf List(Middle) < SearchItem Then 09 Low = Middle + 1 10 Else 11 High = Middle - 1 12 End If 13 Loop 14 BinarySearch = Middle 15 End Function

The programming environment reported a syntax error in the `BinarySearch` code.

State the line number:

Write the correct code for this line.

..... [2]

- (b) (i) State whether programs written in your programming language are compiled or interpreted.

.....

[1]

- (ii) A programmer corrects the syntax error and tests the function. It does not perform as expected when the search item is not in the list.

State the type of error:

Write down the line number where the error occurs.

.....

Write the correct code for this line.

..... [2]

- (iii) State the programming environment you have used when debugging program code.

.....

.....

Name **two** debugging features and describe how they are used.

1

.....

.....

2

.....

.....

[4]

- 3 The following table shows part of the instruction set for a processor which has one general purpose register, the Accumulator (ACC), and an index register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the given address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n into IX.
STO	<address>	Store the contents of ACC at the given address.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

A programmer is writing a program that outputs a string, first in its original order and then in reverse order.

The program will use locations starting at address NAME to store the characters in the string. The location with address MAX stores the number of characters that make up the string.

The programmer has started to write the program in the table opposite. The Comment column contains descriptions for the missing program instructions.

Complete the program using op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// initialise index register to zero
			// initialise COUNT to zero
LOOP1:			// load character from indexed address NAME
			// output character to screen
			// increment index register
			// increment COUNT starts here
			// is COUNT = MAX ?
			// if FALSE, jump to LOOP1
REVERSE:			// decrement index register
			// set ACC to zero
			// store in COUNT
LOOP2:			// load character from indexed address NAME
			// output character to screen
			// decrement index register
			// increment COUNT starts here
			// is COUNT = MAX ?
			// if FALSE, jump to LOOP2
			// end of program
COUNT:			
MAX:	4		
NAME:	B01000110		// ASCII code in binary for 'F'
	B01010010		// ASCII code in binary for 'R'
	B01000101		// ASCII code in binary for 'E'
	B01000100		// ASCII code in binary for 'D'

[15]

- 4** Commercial software usually undergoes acceptance testing and integration testing.

Distinguish between the two types of testing by stating:

- who does the testing
- when the testing occurs
- the specific purpose of each type of testing

(i) Acceptance testing

Who

.....

When

.....

Purpose

..... [3]

(ii) Integration testing

Who

.....

When

.....

Purpose

..... [3]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

CANDIDATE
NAME

--	--	--	--	--

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

October/November 2021

2 hours

You must answer on the question paper.

No additional materials are needed.

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.

This document has **20** pages. Any blank pages are indicated.

- 1 Sandy is writing a program to process data in a stack. The stack is implemented as a 1D array, DataStack, which has up to 100 elements.

The function Push (Value) stores Value on the stack and returns TRUE if Value was added to the stack, or FALSE if the stack is full.

The function Pop () returns the item at the top of the stack, or returns -1 if the stack is empty.

DataStack and TopPointer are declared as global.

- (a) Show the state of DataStack and its pointer after the following functions are executed on the current contents.

Pop ()

Pop ()

Push (19)

Pop ()

Push (50)

TopPointer	3	Index	Data
		[7]	
		[6]	
		[5]	
		[4]	
		[3]	8
		[2]	6
		[1]	20
		[0]	10
			[2]

- (b) Write **program code** for the function `Pop()`.

Programming language

Program code

(c) Sandy has also used a queue in her program.

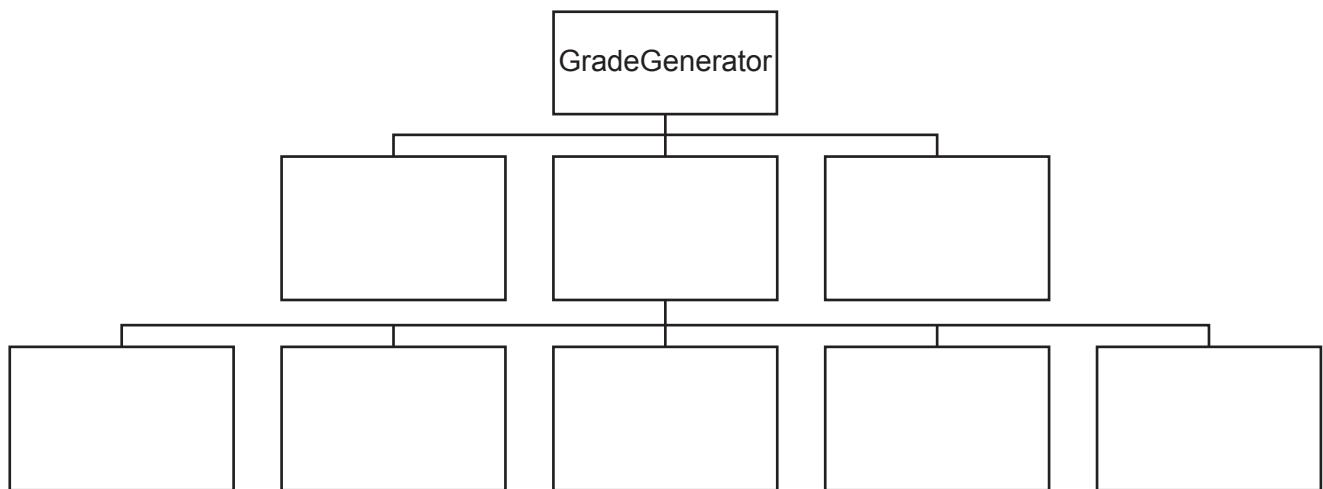
Describe the ways in which a queue differs from a stack.

.....
.....
.....
.....

- 2** A grade generator program takes the mark a student obtained in a test as input.

The program calculates and outputs the grade that matches the mark. The grade is either A, B, C, D or U.

Complete the following JSP structure diagram for the grade generator program.



[4]

- 3 The following pseudocode algorithm performs a binary search on the sorted array ThisArray.

The algorithm returns either the location of SearchItem in the array, or -1 if SearchItem is not in the array.

The function DIV returns the integer value of the division, for example, 11 DIV 2 returns 5.

Complete the algorithm by writing the missing pseudocode statements.

```

FUNCTION BinarySearch(ThisArray[], LowerBound, UpperBound,
                      SearchItem : INTEGER) RETURNS INTEGER

DECLARE Flag : BOOLEAN
DECLARE Mid : INTEGER
Flag ← -2
WHILE Flag <> -1
    Mid ← LowerBound + ((UpperBound - LowerBound) DIV 2)
    IF ..... < .....
        THEN
            RETURN .....
        ELSE
            IF ThisArray[Mid] > SearchItem
                THEN
                    UpperBound ← Mid .....
                ELSE
                    IF ThisArray[Mid] < SearchItem
                        THEN
                            LowerBound ← Mid .....
                        ELSE
                            RETURN .....
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDWHILE
ENDFUNCTION

```

[6]

- 4 Teachers in a school may work on Mondays, Tuesdays and Wednesdays. There are three time slots on each day: time slot 1, time slot 2 and time slot 3.

A teacher is either busy or free.

The school is using a declarative language to write a program to record which teachers are busy in each time slot on each day.

The following knowledge base is used:

```

01 teacher(james).
02 teacher(jill).
03 teacher(karl).
04 teacher(kira).
05 day(monday).
06 day(tuesday).
07 day(wednesday).
08 timeSlot(1).
09 timeSlot(2).
10 timeSlot(3).
11 busy(james, monday, 1).
12 busy(james, tuesday, 2).
13 busy(karl, monday, 1).
14 busy(kira, wednesday, 3).
```

These clauses have the following meaning:

Clause	Explanation
01	James is a teacher
05	Monday is a day
08	1 is a time slot
11	James is busy in time slot 1 on Monday

- (a) More facts need to be included.

Fred is a teacher who is busy in time slot 1 on Tuesday.

Write additional clauses for these facts.

15

16

[2]

- (b) Additional clauses are needed to identify whether Jill is busy in time slot 1 on Monday, Tuesday, or Wednesday.

Write these additional clauses.

17

18

19

[2]

- (c) Write a goal, using the variable x , to find all the teachers who are busy in time slot 3 on Monday.

.....

..... [1]

- (d) Write a rule to find whether a teacher x is free in a specific time slot y on day z .

IsTeacherFree(X, Z, Y)

IF

.....

.....

..... [4]

- 5 The recursive algorithm for the `Recursion()` function is defined in pseudocode as follows:

```

FUNCTION Recursion(A, B : INTEGER) RETURNS INTEGER

IF A <= 100

THEN

RETURN 1

ELSE

IF A > B

THEN

RETURN 5 + Recursion(A - 1, B)

ELSE

RETURN 10 + Recursion(A - 10, B)

ENDIF

ENDIF

ENDFUNCTION

```

- (a) The function is called with the following pseudocode statement:

```
OUTPUT Recursion(104, 102)
```

Dry run the function and complete the trace table. Give the output the program will produce.

Trace table:

Function call	A	B	Return value

Output =

Working
.....
.....

[4]

- (b) Rewrite the function `Recursion()` in **pseudocode**, using an **iterative algorithm**.

[4]

- 6 Kobi is writing an application that uses a record structure to store data.

- (a) (i) Describe what is meant by a **record structure**.

.....

 [2]

- (ii) The record structure stores the unique ID number (a whole number), first name and last name of a customer.

Write a **pseudocode** declaration for the record structure CustomerData.

.....

 [2]

- (b) Kobi's application stores the records in a random access file.

The function `StoreRecord()`:

- takes a customer record as a parameter
- uses the function `CustomerHash()` to calculate and return the hash value for its parameter
- stores the customer record in the returned hash value address.

Assume there are no collisions.

Complete the following pseudocode algorithm to write a new record to the random access file.

```
PROCEDURE StoreRecord(NewData : .....)

    HashValue ← CustomerHash(NewData.CustomerID)

    Filename ← "CustomerRecords.dat"

    OPENFILE Filename FOR .....

    SEEK Filename, .....

    PUTRECORD Filename, .....

    ..... Filename

ENDPROCEDURE
```

[5]

- (c) Identify **two** typical features of a debugger **and** describe how Kobi could use each one during the development of the application.

Feature 1

.....
.....
.....

Feature 2

.....
.....
.....

[4]

- (d) Give **one** benefit and **one** drawback of Kobi using a program generator whilst developing his application.

Benefit

.....

Drawback

[2]

- 7 Sonya is writing a computer program that requires a user input. The user should input an integer between 1 and 100. Sonya wants to use exception handling.

- (a) Explain the reasons why Sonya should use exception handling in her program.

.....
.....
.....
..... [2]

- (b) Write **program code** to read in the number from the user and raise an exception if the data is not valid.

Programming language

Program code

.....
.....
.....
.....
.....
.....
.....
..... [3]

- (c) Give **two other** examples of where exception handling can be used in a program.

1

2

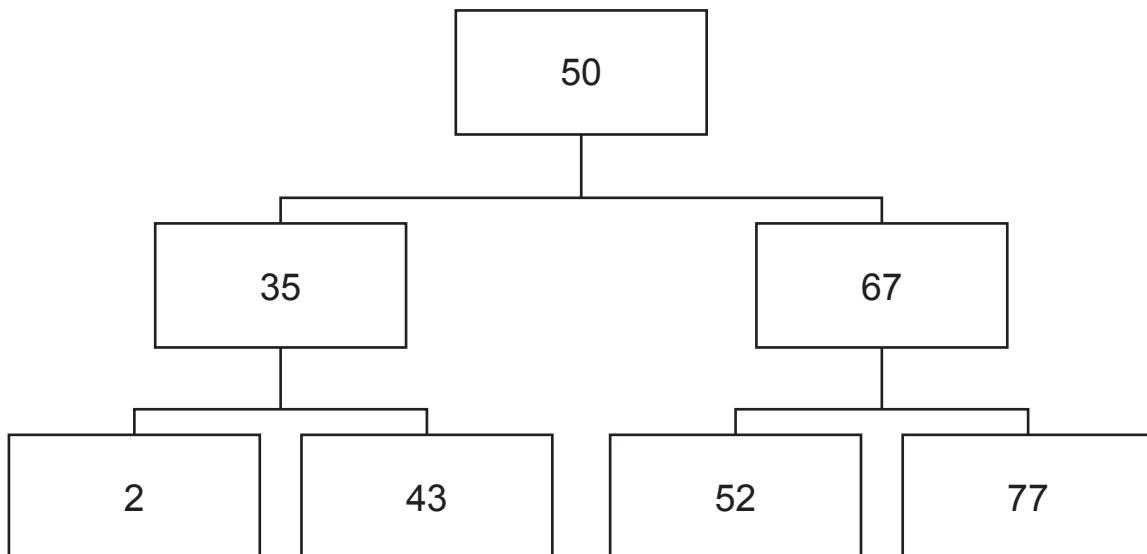
[2]

- 8 Data entered into a computer is stored in an ordered binary tree.

The binary tree is stored in a 2D array, `BinaryTree`.

The first element of the array is index 0.

- (a) The current contents of the binary tree are:



Complete the `LeftPointer` and `RightPointer` values in the following table for the binary tree shown.

A null pointer is represented by -1.

RootNode	0	Index	LeftPointer	Data	RightPointer
[0]				50	
[1]				67	
[2]				77	
[3]				35	
[4]				2	
[5]				43	
[6]				52	
[7]					
[8]					
[9]					
[10]					

[2]

- (b) A post-order tree traversal outputs the left node, then the right node, then the root node.

In the tree given in **part (a)**, the post-order tree traversal would output:

2 43 35 52 77 67 50

Complete the following recursive pseudocode algorithm `PostOrder()`.

```
PROCEDURE PostOrder (..... : INTEGER)

IF BinaryTree[RootNode, 0] <> -1

THEN
    ..... (BinaryTree[RootNode, .....])

ENDIF

IF BinaryTree[RootNode, 2] <> -1

THEN
    ..... (BinaryTree[RootNode, 2])

ENDIF

OUTPUT BinaryTree[RootNode, .....]

ENDPROCEDURE
```

[5]

- 9 A program uses a hashing algorithm to store data in the global array, `StoredData`.

The first element of the array is index 0. The array has 10000 integer elements.

- (a) Write a **pseudocode** declaration for the array `StoredData` **and** initialise each element to `-1`.

[3]

[3]

- (b) The hashing algorithm calculates the remainder after dividing the data by 1000, and then adds 6 to it.

The function `AddItem()` takes the data as a parameter. It calculates the index to store the data using the hashing algorithm.

If there is a collision, the function:

- checks the next index until it finds an index that does not have data in it
 - continues to search from the start of the array, if it reaches the end of the array.

The function returns `TRUE` if the item was successfully added, and `FALSE` if the array is full.

Write **program code** for the function AddItem().

Programming language

Program code

[7]

[7]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

CANDIDATE
NAME

--	--	--	--	--

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

May/June 2021

2 hours

You must answer on the question paper.

No additional materials are needed.

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.

This document has **20** pages. Any blank pages are indicated.

- 1 A vending machine allows users to insert coins to purchase an item.

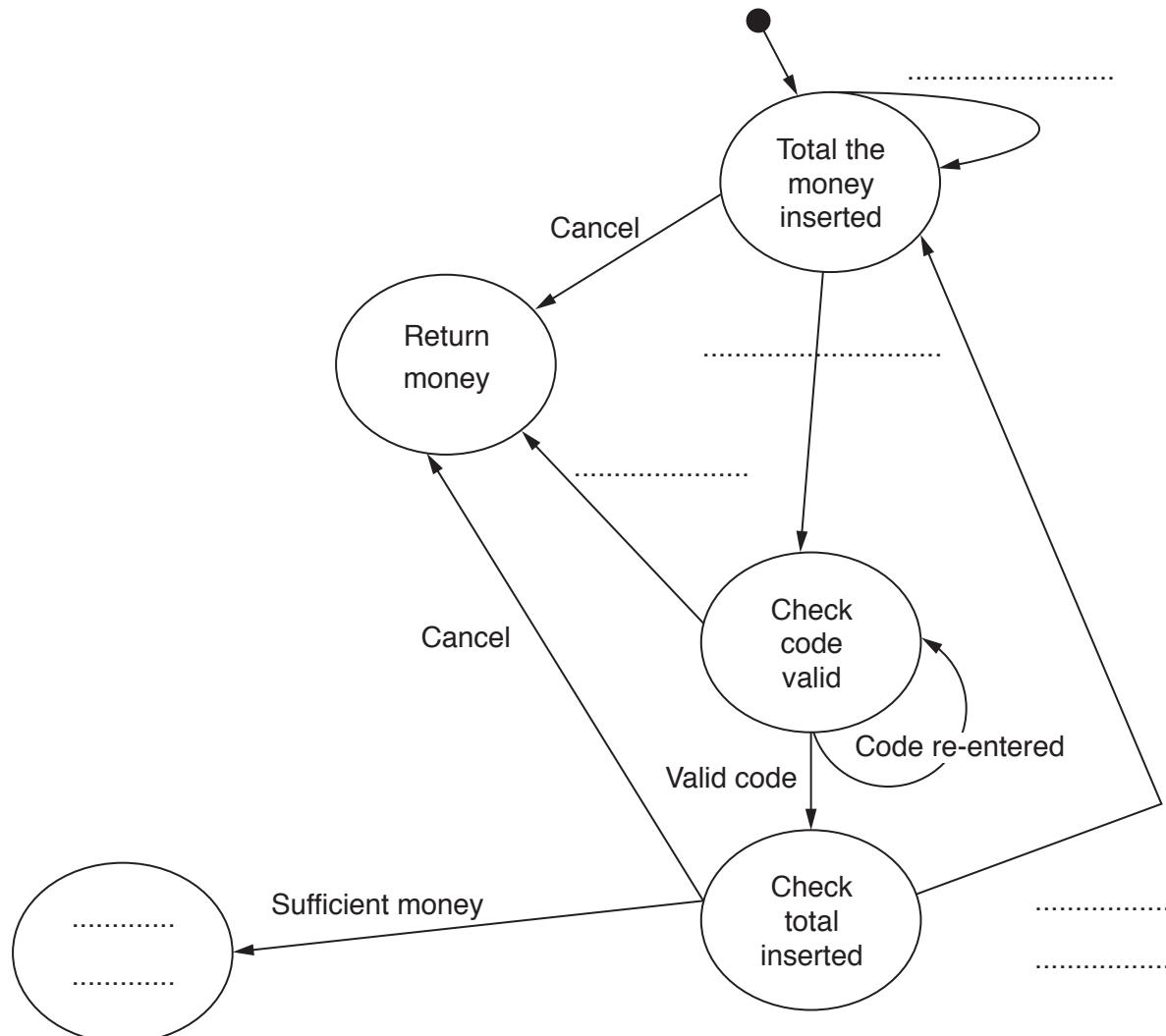
The user then enters the code for the item they would like the machine to dispense (give out). The user must re-enter the code until it is valid.

If the code is valid but the user has not inserted enough money for the item chosen, the machine waits for more coins to be inserted. The user then has to re-enter the code.

The user can press cancel at any time to return the money inserted into the machine.

- (a) The state-transition diagram shows the different states of the vending machine.

Complete the state-transition diagram.



[5]

- (b) The vending machine is part of a program that is written using object-oriented programming (OOP). The vending machine makes use of two classes that are described in the following tables.

All attributes are declared as private.

foodItem	
name : STRING	// the name of the item of food
code : STRING	// the code to be entered for that item to be // selected
cost : REAL	// the cost of the item
constructor(nameP, codeP, costP)	// creates an instance of foodItem // takes the name, code and cost as parameters
getCode() getCost() getName()	// returns the code for the item // returns the cost of the item // returns the name of the item

vendingMachine	
items : ARRAY[0:3] OF foodItem moneyIn : REAL	// stores four items of type foodItem // stores the total money inserted by the // user, initialised to 0 in the constructor
constructor(item1, item2, item3, item4)	// creates an instance of vendingMachine, // takes four objects of type foodItem as // parameters and stores them in array items
insertMoney()	// takes the value of the coin as a parameter // and adds it to moneyIn
checkValid ()	// takes a code as a parameter and checks it is // valid against the food item codes
getItemName()	// takes the array index as a parameter and // returns the name of the food items

- (i) Write **program code** to declare the class vendingMachine. You are only required to write program code for the attribute declarations and the constructor.

If you are writing in Python, include attribute declarations using comments.

Use your programming language's constructor method.

Programming language

Program code

[4]

[4]

- (ii) The method `checkValid()` takes the food item code as a parameter. It checks the code against each element in `items` and returns:

- -1 if the code is not valid
 - -2 if the code is valid, but the `moneyIn` is less than the cost of the item
 - the index of the item, if the code is valid and the `moneyIn` is greater than or equal to the cost of the item.

Write **program code** for the method `checkValid()`.

Programming language

Program code

[5]

- (iii) Four objects of type `foodItem` are declared with the identifiers:

chocolate, sweets, sandwich, apple

Write **program code** to declare an instance of vendingMachine with the identifier machineOne and the objects: chocolate, sweets, sandwich, apple.

Programming language

Program code

[2]

[2]

2 Peter uses a record structure, `customer`, to store data about customers. The data includes:

- a unique customer ID between 10 000 and 99 999
- the customer's first name
- the customer's last name
- the customer's telephone number (for example, +44 1234567891).

(a) Write **pseudocode** to define the record type `customer`.

.....

 [3]

(b) The customer records are stored in a random file. The location of each record is calculated as a hash value using:

$$(\text{customer}. \text{customerID} \bmod 1000) + 2$$

(i) Calculate the hash value for each of the customer IDs in the following table.

Customer ID	Hash value
40125	
10131	

[1]

(ii) Two or more records could have the same hash value that results in a collision.

Explain how the hashing algorithm can be designed to handle collisions.

.....

 [3]

(iii) The function, getCustomer():

- takes the customer ID as a parameter
 - passes the customer ID to the function `getRecordLocation()`, which returns the calculated hash value
 - reads and returns the record from the hashed location in the file `customerRecords.dat`

You can assume that both the file and the record being accessed exist.

Write pseudocode for the function `getCustomer()`.

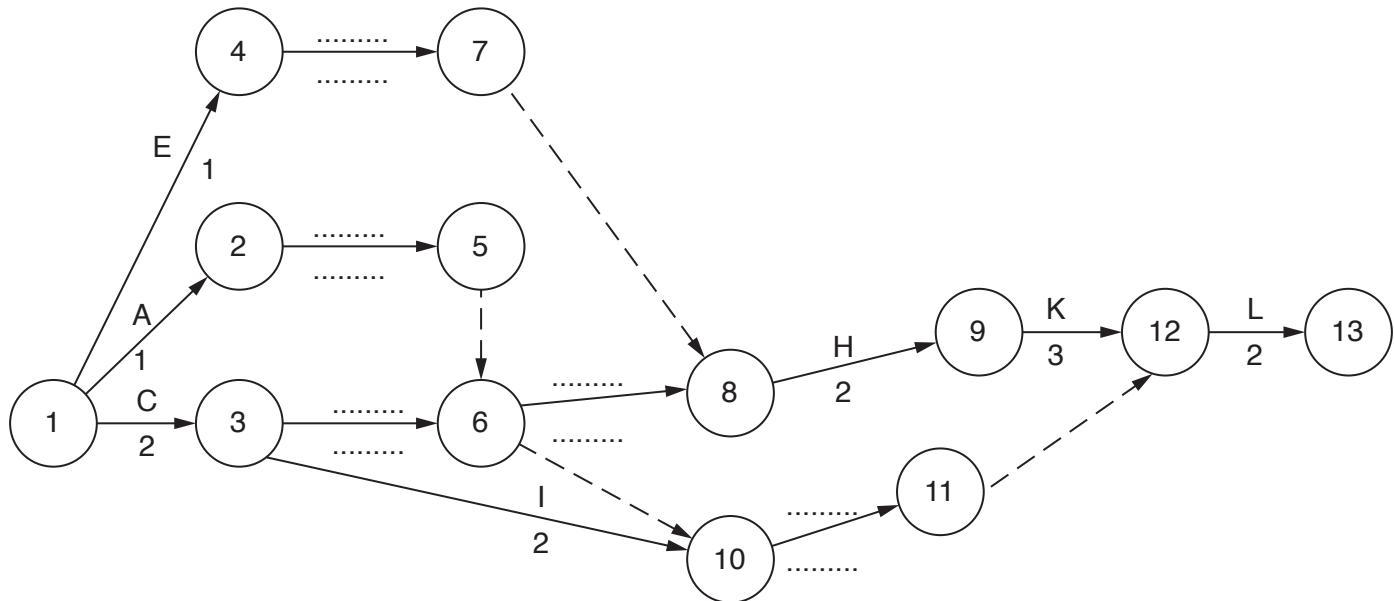
[5]

- 3 Alix manages a team of programmers who are creating a new computer game.

Alix has listed some of the tasks, along with their estimated time to complete and their immediate predecessors in the following table:

Task	Description	Predecessors	Time to complete (weeks)
A	Design character	–	1
B	Program character movement	A	1
C	Design level 1	–	2
D	Program level 1	C	2
E	Design robot	–	1
F	Program robot movement	E	1
G	Integrate character in level 1	B, D	2
H	Integrate robot in level 1	F, G	2
I	Design level 2	C	2
J	Program level 2	D, I	2
K	Test level 1	H	3
L	Integrate character and robot into level 2	J, K	2

- (a) Complete the Program Evaluation Review Technique (PERT) chart for the tasks in the table.



[5]

- (b) Explain how the tasks in the table can be divided between the team to allow concurrency of tasks.

.....
.....
.....
.....
.....
..... [2]

- (c) Explain the benefits of the team using program libraries in the development of the program.

.....
.....
.....
.....
.....
..... [3]

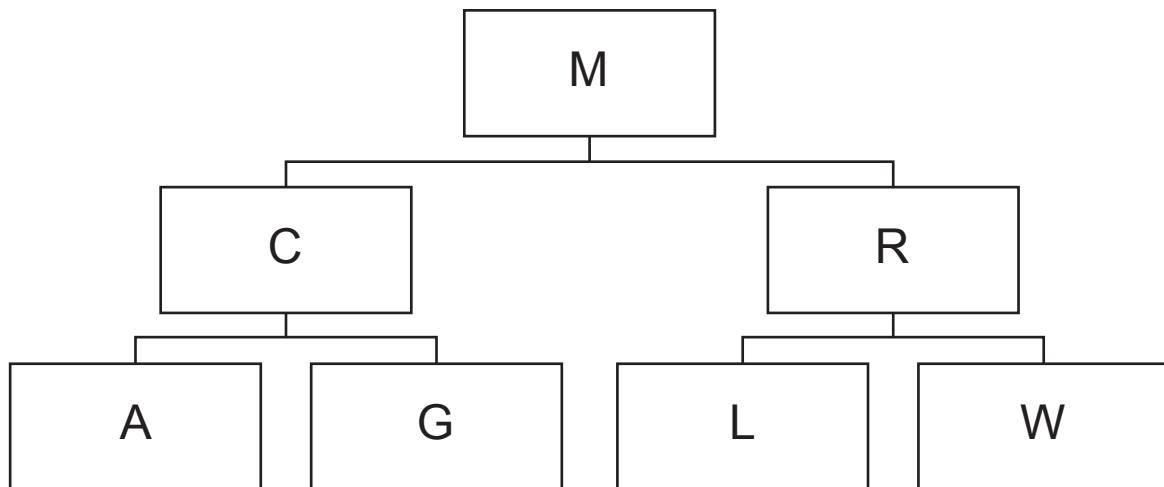
- (d) Identify **two** features in an editor that the developers can use to help them create their programs.

Feature 1

Feature 2

[2]

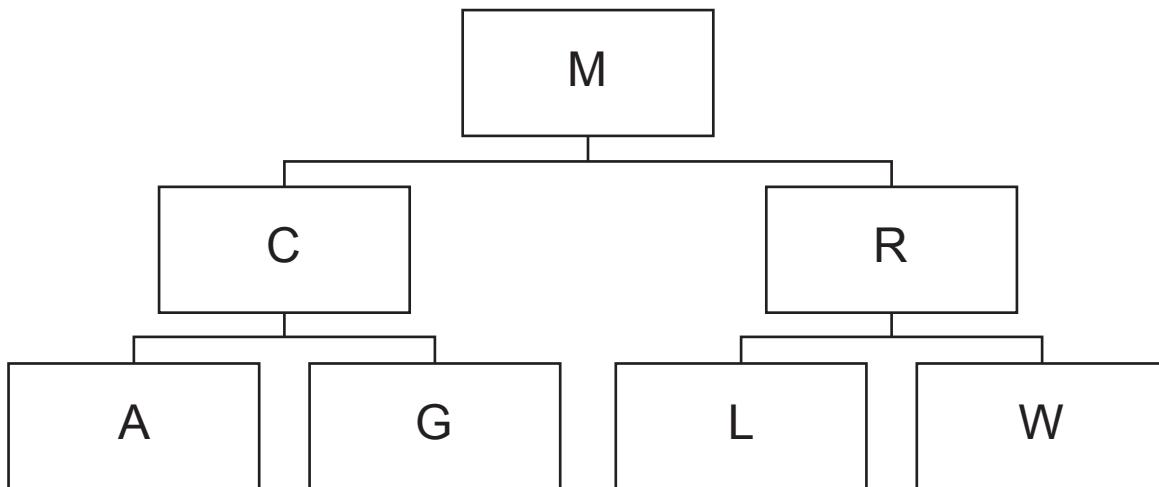
- 4 Chon creates a binary tree structure to store options that the user can select from a menu (M) in his program.



- (a) There are four new options that need to be added.

If option G is selected, the user must choose either option D or option H. If option L is selected, the user must choose either option J or option P.

Complete the following binary tree by adding options D, H, J and P.



- (b) Each node in the binary tree is stored using the following record structure:

```

TYPE node

    leftPointer : INTEGER
    data : STRING
    rightPointer : INTEGER

ENDTYPE

```

The tree is stored as a 1D array, `binaryTree`. Null pointers are represented by `-1`.

- (i) The table shows the contents of the three fields in each record stored in the 1D array `binaryTree`.

Complete the table to show the contents of `binaryTree` from part (a).

<code>rootPointer</code>	<code>freePointer</code>	Index	<code>leftPointer</code>	<code>data</code>	<code>rightPointer</code>
		0		M	
		1		C	
		2		A	
		3		L	
		4		G	
		5		R	
		6		W	
		7		J	
		8		D	
		9		P	
		10		H	
		11			

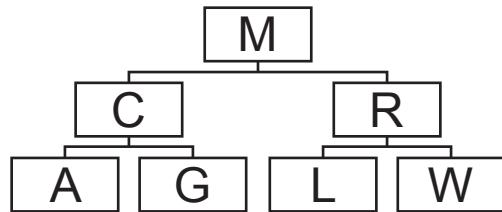
[4]

- (ii) Write **pseudocode** to declare the array `binaryTree` to store up to 100 objects of type `node`.

[2]

[2]

- (iii) A pre-order traversal on the following tree would output M C A G R L W



The pre-order traversal can be written as a recursive procedure:

1. output the root node
 2. follow the left pointer and repeat from step 1
 3. follow the right pointer and repeat from step 1.

Complete the **pseudocode** recursive procedure `preOrder()`.

```
PROCEDURE preOrder(BYVALUE rootPointer : INTEGER)
```

.....

.....
.....
.....
.....
.....
.....
.....
.....

ENDPROCEDURE

[6]

5 A binary search algorithm searches for data in a sorted array.

- (a) The pseudocode function `binarySearch()` performs a binary search to find a given value in the global array, `dataArray`. If the value is found, the function returns its index. If the value is not found, the function returns `-1`.

Complete the **pseudocode** for the function `binarySearch()`.

```

FUNCTION binarySearch(BYVALUE upper, lower, searchValue : INTEGER)
RETURNS INTEGER

DECLARE flag : INTEGER
DECLARE mid : INTEGER
flag ← -2
mid ← 0
WHILE flag <> -1
    mid ← lower + ((upper - lower) ..... )
    IF upper < lower
        THEN
            RETURN .....
    ELSE
        IF dataArray(mid) < searchValue
            THEN
                ..... ← .....
        ELSE
            IF dataArray(mid) > searchValue
                THEN
                    ..... ← .....
            ELSE
                RETURN .....
            ENDIF
        ENDIF
    ENDIF
ENDWHILE
ENDFUNCTION

```

[4]

- (b) The binary search algorithm can be written recursively.

Write **program code** for a recursive function `recursiveBinarySearch()`.

Programming language

Program code

[5]

- 6 The table shows assembly language instructions for a processor that has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

	Instruction		Explanation
Label	Op code	Operand	
	LDM	#n	Immediate addressing. Load the number n to ACC
	LDI	<address>	Direct addressing. Load the contents of the location at the given address to ACC
	LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC
	LDR	#n	Immediate addressing. Load the number n to IX
	STO	<address>	Store contents of ACC at the given address
	ADD	<address>	Add the contents of the given address to ACC
	INC	<register>	Add 1 to the contents of the register (ACC or IX)
	AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
	XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>
	OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>
	OUT		Output to screen the character whose ASCII value is stored in ACC
	CMP	<address>	Compare the contents of ACC with the contents of <address>
	CMP	#n	Compare the contents of ACC with number n
	JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
	JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
	JMP	<address>	Jump to the given address
	END		Return control to the operating system
<label>:	<op code>	<operand>	Labels an instruction
<label>:	<data>		Gives a symbolic address <label> to the memory location with contents <data>

An algorithm takes each letter of a stored 5-letter word and checks if the letter is upper case.

If the letter is upper case, it outputs the letter.

If the letter is not upper case, it converts the letter to upper case and then outputs it.

All ASCII upper case letters have 010 as the three most significant bits.

Assume each letter is alphabetic.

Complete the assembly language program for the algorithm described using the instruction set provided on the previous page.

Instruction			Comment	
Label	Op code	Operand		
	LDR	#0	// load zero to IX	
			// load count and check if it is 5	
	JPE	endP	// jump to end	
	LDX	word	// load letter from indexed address word	
			// check if it is upper case	
	CMP	#0		
	JPE	output	// jump to output if it is upper case	
	LDX	word	// load letter from indexed address word	
			// convert to upper case	
output:	OUT		// output the character	
			// increase count by 1	
	INC	IX	// increase IX by 1	
	JMP	start	// return to start	
endP:	end		// end the program	
word:	B01001000			
	B01101111			
	B01110101			
	B01110011			
	B01100101			
mask1:	B00100000			
mask2:	B11011111			
count:	0			

[6]

- 7 Giles is writing a program that uses a stack.

The stack stores up to 1000 integers in the 1D array, `stackArray`.

- (a) The procedure `setUpStack()` takes two parameters:

- the array, `stackArray`
- a pointer to the last element pushed onto the stack, `topOfStack`

The procedure initialises all array elements to -1 and the pointer to -1 .

Write **pseudocode** for the procedure `setUpStack()`.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [3]

- (b) The function `pop()` pops and returns the item from the top of the stack. If the stack is empty, it returns -1 .

Write **pseudocode** for the function `pop()`.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [3]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/43

Paper 4 Practical

October/November 2022

2 hours 30 minutes

You will need: Candidate source files (listed on page 2)
evidence.doc



INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the evidence document, **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: **evidence_zz999_999**

A class declaration can be used to declare a record.

If the programming language does not support arrays, a list can be used instead.

A source file is used to answer **Question 1**. The file is called **IntegerData.txt**

- 1 The text file `IntegerData.txt` stores 100 integer numbers between 1 and 100 inclusive. A program is required to read in this data and perform searching and sorting on the data.

- (a) Write program code to declare a global 1D array, `DataArray`, with space for 100 integer values.

Save your program as **Question1_N22**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[2]

- (b) The procedure `ReadFile()` must read in the numbers from the text file and store each one in the array. Use appropriate exception handling.

Write program code for the procedure `ReadFile()`.

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[6]

- (c) The function `FindValues()` asks the user to enter a number to search for in the array. The number input must be a whole number between 1 and 100 inclusive. The function then returns the number of times the number input appears in the array.

Write program code for the function `FindValues()`.

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[7]

- (d) (i) Write program code to call `ReadFile()` and `FindValues()` from the main program. The return value from `FindValues()` must be output with an appropriate message.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[3]

- (ii) Test your program using the number 61 as input.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 1(d)(ii)** in the evidence document.

[1]

- (e) The procedure `BubbleSort()` needs to perform a bubble sort on the array and print the contents of the sorted array.

Write program code for the procedure `BubbleSort()` **and** call it from the main program.

Save your program.

Copy and paste the program code into **part 1(e)** in the evidence document.

[4]

- 2 A computer program is being developed that uses a set of cards. The program is written using object-oriented programming.

The program has two classes: Card and Hand.

The methods and attributes of these classes are shown:

Card	
Number : INTEGER	stores the card number from 1 to 5 inclusive
Colour : STRING	stores the card colour: red, blue or yellow
Constructor()	takes a number and colour as parameters and sets the private values to these parameters
GetNumber()	returns the card number
GetColour()	returns the card colour

Hand	
Cards : ARRAY[0:9] OF Card	1D array of type Card
FirstCard : INTEGER	stores the position of the first card in the hand
NumberCards : INTEGER	stores the number of cards in the hand
Constructor()	takes five card objects as parameters, assigns each card to the array Cards[], initialises FirstCard to 0 and NumberCards to 5
GetCard()	takes an index as a parameter and returns the card at that index in the array

- (a) (i) Write program code to declare the class Card, its attributes and constructor.

Do **not** write program code for the get methods.

Use your programming language appropriate constructor.

All attributes must be private. If you are writing in Python, include attribute declarations using comments.

Save your program as **Question2_N22**.

Copy and paste the program code into **part 2(a)(i)** in the evidence document.

[5]

- (ii) Write program code for the class methods GetNumber() and GetColour().

Save your program.

Copy and paste the program code into **part 2(a)(ii)** in the evidence document.

[3]

(iii) The program is tested with the following cards:

Number	Colour
1	red
2	red
3	red
4	red
5	red
1	blue
2	blue
3	blue
4	blue
5	blue
1	yellow
2	yellow
3	yellow
4	yellow
5	yellow

Write program code to declare each of these cards as a variable of type `Card` in the main program.

Save your program.

Copy and paste the program code into **part 2(a)(iii)** in the evidence document.

[2]

- (b) (i) Write program code to declare the class `Hand`, its attributes and constructor.

Do **not** write the get methods.

Use your programming language appropriate constructor.

All attributes must be private. If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into **part 2(b)(i)** in the evidence document.

[6]

- (ii) The get method `GetCard()` takes an index as a parameter and returns the card stored at that index in the array.

Write program code for the method `GetCard()`.

Save your program.

Copy and paste the program code into **part 2(b)(ii)** in the evidence document.

[2]

- (iii) Two players are declared with 5 cards each.

Player 1 has the cards: 1 red, 2 red, 3 red, 4 red, 1 yellow.

Player 2 has the cards: 2 yellow, 3 yellow, 4 yellow, 5 yellow, 1 blue.

Write program code to declare player 1 and player 2 as objects of type `Hand`, with the cards indicated.

Save your program.

Copy and paste the program code into **part 2(b)(iii)** in the evidence document.

[2]

(c) The function `CalculateValue()` takes a player's hand as a parameter and returns a score calculated using the following rules:

- If a card is red, 5 points are added to the player's score.
- If a card is blue, 10 points are added to the player's score.
- If a card is yellow, 15 points are added to the player's score.
- The number of each card in the hand is added to the player's score.

(i) Write program code for the function `CalculateValue()`.
Assume that there are only 5 cards in the player's hand in this function.

Save your program.

Copy and paste the program code into **part 2(c)(i)** in the evidence document.

[6]

(ii) Amend the main program by writing program code to use the function `CalculateValue()` for each of the two players. The player with the highest score wins.

Output an appropriate message to identify the winning player, or if the game was a draw (both players have the same number of points).

Save your program.

Copy and paste the program code into **part 2(c)(ii)** in the evidence document.

[4]

(iii) Test your program.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 2(c)(iii)** in the evidence document.

[1]

- 3 A binary tree consists of nodes. Each node has 3 integer values: a left pointer, data and a right pointer.

The binary tree is stored using a global 2D array.

The pseudocode declaration for the array is:

```
DECLARE ArrayNodes : ARRAY[0:19, 0:2] OF INTEGER
```

For example:

- `ArrayNodes[0, 0]` stores the left pointer for the first node.
- `ArrayNodes[0, 1]` stores the data for the first node.
- `ArrayNodes[0, 2]` stores the right pointer for the first node.

`-1` indicates a null pointer, or null data.

(a) Write program code to:

- declare the global 2D array `ArrayNodes`
- initialise all 3 integer values to `-1` for each node.

Save your program as **Question3_N22**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[3]

(b) The binary tree stores the following values:

Index	Left pointer	Data	Right pointer
0	1	20	5
1	2	15	-1
2	-1	3	3
3	-1	9	4
4	-1	10	-1
5	-1	58	-1
6	-1	-1	-1

`FreeNode` stores the index of the first free element in the array, initialised to 6.

`RootPointer` stores the index of the first node in the tree, initialised to 0.

Amend your program by writing program code to store the given data in `ArrayNodes` **and** initialise the free node and root node pointers.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[2]

- (c) The following recursive pseudocode function searches the binary tree for a given value. If the value is found, the function must return the index of the value. If the value is not found, the function must return -1.

The function is incomplete. There are **four** incomplete statements.

```

FUNCTION SearchValue(Root : INTEGER,
                     ValueToFind : INTEGER) RETURNS INTEGER

    IF Root = -1 THEN
        RETURN -1

    ELSE
        IF ArrayNodes[Root, 1] = ValueToFind THEN
            RETURN .....
        ELSE
            IF ArrayNodes[Root, 1] = -1 THEN
                RETURN -1
            ENDIF
        ENDIF
        ENDIF
        IF ArrayNodes[Root, 1] ..... ValueToFind THEN
            RETURN SearchValue(ArrayNodes[....., 0], ValueToFind)
        ENDIF
        IF ArrayNodes[Root, .....] < ValueToFind THEN
            RETURN SearchValue(ArrayNodes[Root, 2], ValueToFind)
        ENDIF
    ENDFUNCTION

```

Write program code for the function `SearchValue()`.

Save your program.

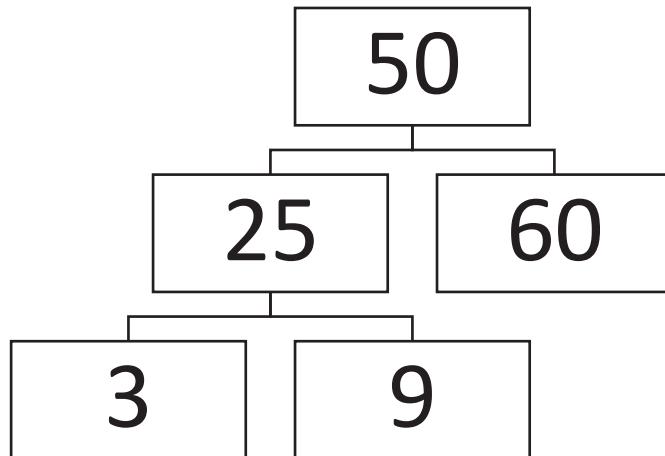
Copy and paste the program code into **part 3(c)** in the evidence document.

[5]

(d) A post order traversal performs the following operation:

- visit the left node
- visit the right node
- output the root.

For example, in the following tree, the output would be: 3 9 25 60 50



An outline of the `PostOrder()` procedure is:

- If left node is not empty, make a recursive call with the left node as the root.
- If right node is not empty, make a recursive call with the right node as the root.
- Output the current root node.

The procedure `PostOrder()` takes the root node as a parameter.

Write program code for the procedure `PostOrder()`.

Save your program.

Copy and paste the program code into **part 3(d)** in the evidence document.

[7]

(e) (i) Amend the main program by writing program code to:

- call the function `SearchValue()` to find the position of the number 15 in the tree
- use the result from `SearchValue()` to output either the index of the value if found, or an appropriate message to state that the value was not found
- call the procedure `PostOrder()`.

Save your program.

Copy and paste the program code into **part 3(e)(i)** in the evidence document.

[3]

(ii) Test your program.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 3(e)(ii)** in the evidence document.

[1]

BLANK PAGE

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.



Cambridge International AS & A Level

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

May/June 2020

2 hours

You must answer on the question paper.

No additional materials are needed.

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.

This document has **16** pages. Blank pages are indicated.

- 1 Carlos is writing exception handling code for his program.

- (a) State what is meant by an **exception**.

..... [1]

- (b) Give **three** situations where an exception handling routine would be required.

1

2

3

[3]

- (c) Describe the benefits of using exception handling in a program.

.....
.....
..... [2]

- 2 (a) Programs can be written using recursion.

Tick (\checkmark) one or more boxes to show the features that **must** be included in a valid recursive algorithm.

Feature	Must be included
Incrementation	
General case	
Base case	
Selection case	
It calls itself	

[2]

- (b) The following recursive procedure outputs every even number from the positive parameter value down to and including 2.

The procedure checks if the integer parameter is an even or an odd number. If the number is odd, the procedure converts it to an even number by subtracting 1 from it.

The function MOD(ThisNum : INTEGER, ThisDiv : INTEGER) returns the remainder value when ThisNum is divided by ThisDiv.

Complete the **pseudocode** for the recursive procedure.

```

PROCEDURE Count (BYVALUE ..... : INTEGER)

    IF ..... (Number, 2) <> 0
        THEN
            Number ← Number - 1
        ENDIF
        OUTPUT .....
        IF Number > 0
            THEN
                ..... ( ..... - 1)
            ENDIF
    ENDPROCEDURE

```

[5]

- (c) A program allows guests to input a meal option at a wedding.

Guests can choose meal option 1 or meal option 2.

The program will keep count of the numbers of each meal option chosen.

The program ends when a value other than 1 or 2 is entered. It then outputs the count of each meal option.

```

PROCEDURE MealsCount (BYREF MealOption1 : INTEGER, MealOption2 : INTEGER)

DECLARE MealOption : INTEGER

DECLARE MoreMeals : BOOLEAN

MoreMeals ← True

WHILE MoreMeals = True

    INPUT MealOption

    IF MealOption = 1

        THEN

            MealOption1 ← MealOption1 + 1

    ELSE

        IF MealOption = 2

            THEN

                MealOption2 ← MealOption2 + 1

            ELSE

                OUTPUT MealOption1, " ", MealOption2

                MoreMeals ← False

            ENDIF

    ENDIF

ENDWHILE

ENDPROCEDURE

```

The program contains a conditional loop.

Use **pseudocode** to rewrite the conditional loop as a recursive algorithm.

```
PROCEDURE MealsCount (BYREF MealOption1 : INTEGER, MealOption2 : INTEGER)
```

```
    DECLARE MealOption : INTEGER
```

```
    .....  
    .....
```

```
    ENDPROCEDURE
```

[5]

- 3 A declarative programming language is used to represent the following knowledge base.

```

01 person(jessica).
02 person(pradeep).
03 person(steffi).
04 person(johann).
05 sport(football).
06 sport(hockey).
07 sport(cricket).
08 sport(volleyball).
09 plays(johann, football).
10 plays(steffi, cricket).
11 plays(jessica, football).
12 will_not_play(pradeep, cricket).

```

These clauses have the following meanings:

Clause	Meaning
01	Jessica is a person
05	Football is a sport
09	Johann plays football
12	Pradeep refuses to play cricket

- (a) Elle is a person who plays rugby but refuses to play hockey.

Write additional clauses to represent this information.

- 13
- 14
- 15
- 16

[4]

- (b) Write the result returned by the goal:

plays(X, football).

X = [1]

- (c) Y might play X, if Y is a person, X is a sport and Y does not refuse to play X.

Write this as a rule.

mightplay(Y , X)

IF

.....

..... [5]

- 4 Object-oriented programming has several features. These include inheritance, classes, methods and properties.

- (a) Describe what is meant by **inheritance**.

.....

.....

.....

..... [2]

- (b) Identify **two other** features of object-oriented programming.

1

2

[2]

- 5 A tennis club is developing a program to store details of the lessons it offers. The programmer has designed the class Lesson for the details of the lessons.

The following class diagram shows the design for the Lesson class.

Lesson	
LessonType : STRING	// initialised in constructor to the parameter // value passed to the constructor
Instructor : STRING	// initialised in constructor to the parameter // value passed to the constructor
Constructor()	// method used to create and initialise an // object
GetLessonType()	// returns LessonType value
GetInstructor()	// returns Instructor value
GetFee()	// returns the cost of a lesson
SetLessonType()	// sets the LessonType to the parameter value
SetInstructor()	// sets the Instructor to the parameter value

- (a) Write **program code** for the Constructor() method.

Use the appropriate constructor method for your chosen programming language.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

[3]

- (b) Write **program code** for the `GetLessonType()` method.

Programming language

Program code

.....
.....
.....
.....
.....
.....

[2]

- (c) Fee is the cost that a customer will pay for a lesson.

The method `GetFee()` validates the parameter value. The method is sent a parameter value that represents the skill level of the customer: beginner, intermediate or advanced.

The parameter will be a character:

- 'B' for beginner
 - 'I' for intermediate
 - 'A' for advanced.

The method must check the parameter value is a valid character ('B', 'I' or 'A') and return the correct fee. It must return -1 if it is not a valid character.

The fees are:

- \$45 for a beginner
 - \$50 for an intermediate
 - \$55 for an advanced.

Write program code for the GetFee () method.

Programming language

Program code

[5]

- (d) The tennis club only offers nine different types of lesson. The lesson objects are stored in a 1D array.

Write **pseudocode** to declare an array `LessonArray` to store the nine lesson objects.

.....
..... [2]

- (e) The tennis club has the lesson ‘Improve Your Serve’ that has David as the instructor.

Write **program code** to create the lesson ‘Improve Your Serve’ as an instance of the class `Lesson`. The object needs to be stored in the third element of the array `LessonArray`.

Programming language

Program code

.....
.....
..... [3]

- 6 A theatre company stores customer login details to allow customers to book tickets online.

A hash table stores login details for 2000 customers.

Each customer's details are stored in a record.

The declaration for CustomerRecord is:

```
TYPE CustomerRecord
  DECLARE UserID : STRING
  DECLARE PINNumber : INTEGER
ENDTYPE
```

A 1D array, CustomerDetails, is used to implement the hash table. CustomerDetails is a global array. The 1D array has 6000 elements.

- (a) The procedure InitialiseHashTable() initialises the hash table. UserID is initialised as an empty string, and PINNumber initialised to 0 for all of the records.

Write **pseudocode** for the procedure InitialiseHashTable().

.....

 [4]

- (b) The function InsertRecord() is used to insert a new record into the hash table.

The function, Hash():

- takes a UserID as a parameter
- performs the hashing algorithm
- returns the calculated index of the user ID within the hash table.

If the hash table is full, the function InsertRecord() returns -1. If there is space available in the hash table, the record is inserted, and it returns the position of this record in the array.

Complete the **pseudocode** for the function.

```

FUNCTION InsertRecord(NewRecord : CustomerRecord) RETURNS INTEGER

DECLARE Count : INTEGER
DECLARE Index : INTEGER
Count ← 0
Index ← Hash(.....)
WHILE (CustomerDetails[Index].UserID <> "") ..... (Count <= 5999)

    Index ← Index + 1
    Count ← Count + 1
    IF Index > 5999
        THEN
            .....
    ENDIF
ENDWHILE
IF Count > 5999
    THEN
        .....
ELSE
    CustomerDetails[.....] ← .....
    .....
ENDIF
ENDFUNCTION

```

[7]

- 7 (a) A shirt design company has an order form to order shirts. Customers can order multiple shirts using the same form.

The customer details section has the data:

- name
- address
- telephone number.

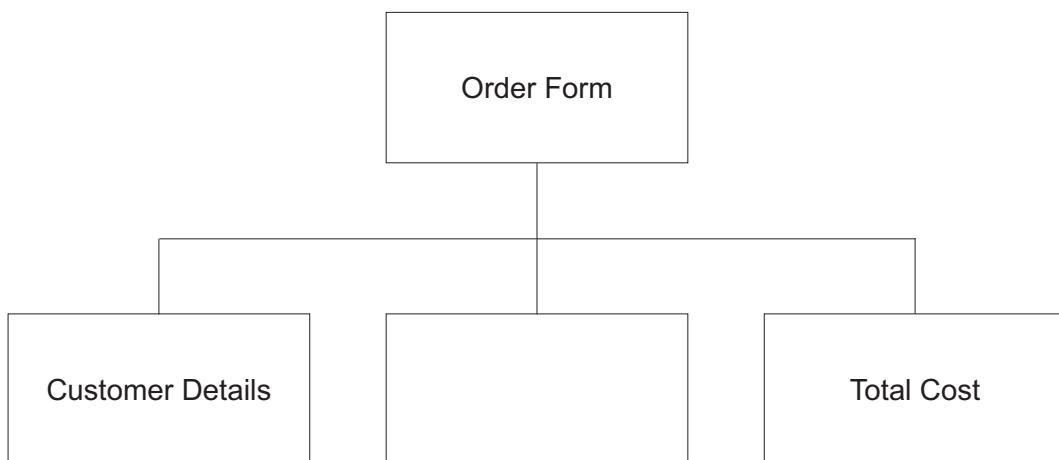
The order details section has the data:

- shirt ID
- colour
- cost.

A total cost for the order is also calculated.

The cost of each shirt is dependent on the size ordered. The sizes customers can order are small, medium and large.

Complete the following JSP data structure diagram for the order form.



[7]

- (b) Each customer's order is stored as a record in a file. The customers' orders are stored in the order in which they arrive in the file and no key field is used.

(i) Identify this type of file structure.

..... [1]

(ii) Identify **two other** types of file structure.

1

2

[2]

- (c) The procedure `UpdateTelephone()` allows the shirt company to update the record for a customer's details. The procedure will update the telephone number.

The program stores customer details as a custom data type, `Customer`.

The definition for this data type is:

```
TYPE Customer
```

```
    Name : STRING
```

```
    Address : STRING
```

```
    TelephoneNumber : STRING
```

```
ENDTYPE
```

The procedure `UpdateTelephone()` takes the customer record to be updated and the new telephone number as parameters. It then updates the telephone number in the record.

Complete the **pseudocode** for the procedure `UpdateTelephone()`.

```
PROCEDURE UpdateTelephone (..... ThisCustomer : Customer,  
..... NewTelephoneNumber : STRING)
```

```
.....
```

[3]

(d) The shirt company is looking to implement a system to reward customers. The system includes:

- 10% discount on orders over \$50
- free gift if order over \$50 and if the order is placed on a Monday
- additional 5% discount if a customer has a loyalty card
- free delivery for a customer with a loyalty card and spends over \$50.

Complete the following decision table for this system.

Conditions	Order over \$50	Y	Y	Y	Y	N	N	N	N
	Monday	Y	Y	N	N	Y	Y	N	N
	Loyalty card	Y	N	Y	N	Y	N	Y	N
	Additional 5% discount								
	10% discount								
	Free gift								
	Free delivery								

[4]

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/43

Paper 4 Practical

May/June 2022

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the document **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

A source file is used to answer question 1. The file is called **HighScore.txt**

- 1 The text file HighScore.txt stores the players who have scored the top ten scores in a game, in descending order of score. The file stores the 3-character name of the player, and their integer score, in the order: player, score.

For example, the current top player in the text file:

FYI is the player name

10 000 is the score

The program:

- reads in the data from HighScore.txt
- allows the user to enter a new player name and their score
- if appropriate, inserts the new player (name and score) into the top ten
- writes the top ten players (name and score) into a new text file NewHighScore.txt

- (a) The program stores the players and their scores in an array of 11 elements (10 elements to be read from the file, 1 element to be inserted by the user).

Write a program to declare one or more arrays, as global data structures, to store the player names and their scores.

Save your program as **Question1_J2022**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[2]

- (b) The procedure ReadHighScores () opens the file HighScore.txt and reads the data into the data structure(s) declared in **part 1(a)**.

Write program code to declare the procedure ReadHighScores () .

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[6]

- (c) The procedure `OutputHighScores()` outputs all the values in the data structure(s) in the format:

PlayerName Score

For example, the first two data items: FYI 10000
ABC 9092

Write program code to declare the procedure `OutputHighScores()`.

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[3]

- (d) The main program should first call `ReadHighScores()` and then `OutputHighScores()`.

- (i) Write the program code for the main program.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[2]

- (ii) Test your program.

Take a screenshot to show the output from **part 1(d)(i)**.

Copy and paste the screenshot into **part 1(d)(ii)** in the evidence document.

[1]

- (e) The main program needs to ask the user to input a new player name and a score. If this score is in the top ten then it will create a new top ten list that includes this score.

- (i) Amend the main program to ask the user to input a 3-character player name and an integer score that must be between 1 and 100 000 inclusive.

Save your program.

Copy and paste the program code into **part 1(e)(i)** in the evidence document.

[3]

- (ii) Write program code to declare a procedure that:

- takes the player name and score as parameters
- creates a new top ten list that includes the parameter if appropriate.

Save your program.

Copy and paste the program code into **part 1(e)(ii)** in the evidence document.

[5]

- (iii) Amend the main program to call the procedure from **part 1(e)(ii)**.

Output the contents of the array before inserting the new player name and score, and output the contents of the array after inserting the new player name and score.

Save your program.

Copy and paste the program code into **part 1(e)(iii)** in the evidence document.

[2]

- (iv) Test your program by entering the player name "JKL" and the score "9999".

Take a screenshot to show the output.

Copy and paste the screenshot into **part 1(e)(iv)** in the evidence document.

[1]

- (f) The procedure `WriteTopTen()` stores the new top ten player names and scores in a text file called `NewHighScore.txt`

Write program code to declare the procedure `WriteTopTen()`.

Save your program.

Copy and paste the program code into **part 1(f)** in the evidence document.

[4]

- 2 A computer game is being developed using object-oriented programming.

One element of the game is a balloon. This is designed as the class `Balloon`.

The class has the following attributes and methods.

Balloon	
Health : INTEGER	The health of the balloon
Colour : STRING	The colour of the balloon
DefenceItem : STRING	The item the balloon uses to defend itself
Constructor()	Initialises the defence item and colour using the parameters Initialises health to 100
ChangeHealth()	Takes the change as a parameter and adds this to the health
GetDefenceItem()	Returns the defence item of the object
CheckHealth()	If the health is 0 or less, it returns TRUE, otherwise it returns FALSE

- (a) The constructor takes the name of the defence item and the balloon's colour as parameters and sets these to the attributes. The health is initialised to 100.

Write program code to declare the class `Balloon` and its constructor. Do not write any other methods.

Use your language appropriate constructor.

All attributes should be private. If you are writing in Python include attribute declarations using comments.

Save your program as **Question2_J2022**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[5]

- (b) The get method `GetDefenceItem()` returns the defence item of the object.

Amend your program code to include the get method `GetDefenceItem()`.

Save your program.

Copy and paste the program code into **part 2(b)** in the evidence document.

[2]

- (c) The object's method `ChangeHealth()` takes an integer number as a parameter and adds this to the health attribute of the object.

Amend your program code to include the method `ChangeHealth()`.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[2]

- (d) The object's method `CheckHealth()` returns `TRUE` if the health of the object is 0 or less (no health remaining) and returns `FALSE` otherwise (health remaining).

Amend your program code to include the method `CheckHealth()`.

Save your program.

Copy and paste the program code into **part 2(d)** in the evidence document.

[2]

- (e) Amend the main program to:

- take as input a defence item and colour from the user
- create a new balloon with the identifier `Balloon1` using the data input.

Save your program.

Copy and paste the program code into **part 2(e)** in the evidence document.

[3]

- (f) The function `Defend()`:

- takes a balloon object as a parameter
- takes as input the strength of an opponent from the user
- uses the `ChangeHealth()` method to subtract the strength input from the object's health
- outputs the defence item of the balloon
- checks the health of the object and outputs an appropriate message if it has no health remaining, or if it has health remaining
- returns the amended balloon object.

Write program code to declare the function `Defend()`.

Save your program.

Copy and paste the program code into **part 2(f)** in the evidence document.

[8]

(g) (i) Amend the main program to call the function Defend().

Save your program.

Copy and paste the program code into **part 2(g)(i)** in the evidence document.

[2]

(ii) Test your program using the following inputs:

- balloon defence method "Shield"
- balloon colour "Red"
- strength of opponent 50

Take a screenshot to show the output.

Copy and paste the screenshot into **part 2(g)(ii)** in the evidence document.

[1]

- 3 A program uses a circular queue to store strings. The queue is created as a 1D array, QueueArray, with 10 string items.

The following data is stored about the queue:

- the head pointer initialised to 0
- the tail pointer initialised to 0
- the number of items in the queue initialised to 0.

- (a) Declare the array, head pointer, tail pointer and number of items.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_J2022**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[2]

- (b) The function `Enqueue` is written in pseudocode. The function adds `DataToAdd` to the queue. It returns `FALSE` if the queue is full and returns `TRUE` if the item is added.

The function is incomplete, there are **five** incomplete statements.

```

FUNCTION Enqueue (BYREF QueueArray[] : STRING, BYREF HeadPointer : INTEGER,
                  BYREF TailPointer : INTEGER, NumberItems : INTEGER,
                  DataToAdd : STRING) RETURNS BOOLEAN

IF NumberItems = ..... THEN
    RETURN .....
ENDIF

QueueArray[.....] ← DataToAdd

IF TailPointer >= 9 THEN
    TailPointer ← .....
ELSE
    TailPointer ← TailPointer + 1
ENDIF

NumberItems ← NumberItems .....
RETURN TRUE

ENDFUNCTION

```

Write program code for the function `Enqueue ()`.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[7]

- (c) The function `Dequeue ()` returns "FALSE" if the queue is empty, or it returns the next data item in the queue.

Write program code for the function `Dequeue ()`.

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[6]

(d) (i) Amend the main program to:

- take as input 11 string values from the user
- use the `Enqueue()` function to add each element to the queue
- output an appropriate message to state whether each addition was successful, or not
- call `Dequeue()` function twice and output the return value each time.

Save your program.

Copy and paste the program code into **part 3(d)(i)** in the evidence document.

[5]

(ii) Test your program with the input data:

"A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K"

Take a screenshot to show the output.

Copy and paste the screenshot into **part 3(d)(ii)** in the evidence document.

[1]

BLANK PAGE

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.

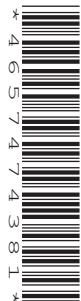
CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

October/November 2019

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

- 1 Each student at CIE University needs a printing account to print documents from university computers.

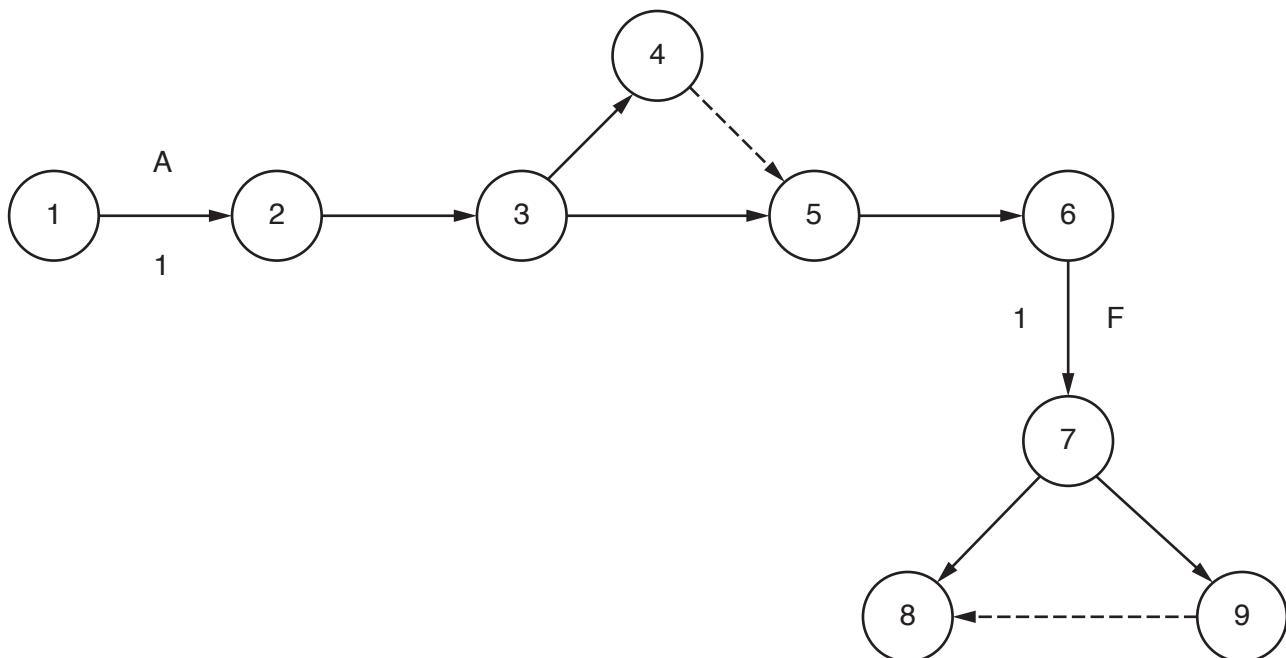
The university is developing software to manage each student's printing account and the printing process.

- (a) Developing the software will include the following activities.

Activity	Description	Time in weeks	Predecessor
A	Identify requirements	1	-
B	Produce design	3	A
C	Write code	10	B
D	Test modules	7	B
E	Final system black-box testing	3	C, D
F	Install software	1	E
G	Acceptance testing	2	F
H	Create user documentation	2	F

- (i) Add the correct activities and times to the following Program Evaluation Review Technique (PERT) chart for the software development.

Two activities and times have been done for you.



[6]

- (ii) State what is meant by the **critical path** in a PERT chart.

.....
.....

[1]

- (iii) Identify **and** describe a project planning technique, other than a PERT chart.

.....
.....
.....

[2]

- (b) When a student prints a document, a print job is created. The print job is sent to a print server.

The print server uses a queue to hold each print job waiting to be printed.

- (i) The queue is circular and has six spaces to hold jobs.

The queue currently holds four jobs waiting to be printed. The jobs have arrived in the order A, B, D, C.

Complete the diagram to show the current contents of the queue.



[1]

- (ii) Print jobs A and B are now complete. Four more print jobs have arrived in the order E, F, G, H.

Complete the diagram to show the current contents and pointers for the queue.



[3]

- (iii) State what would happen if another print job is added to the queue in the status in part (b)(ii).

.....
.....

[1]

- (iv) The queue is stored as an array, Queue, with six elements. The following algorithm removes a print job from the queue and returns it.

Complete the following **pseudocode** for the function Remove.

```

FUNCTION Remove RETURNS STRING
    DECLARE PrintJob : STRING
    IF ..... = EndPointer
        THEN
            RETURN "Empty"
        ELSE
            PrintJob ← Queue[.....]
            IF StartPointer = .....
                THEN
                    StartPointer ← .....
                ELSE
                    StartPointer ← StartPointer + 1
                ENDIF
            RETURN PrintJob
        ENDIF
    ENDFUNCTION

```

[4]

- (v) Explain why the circular queue could not be implemented as a stack.

.....

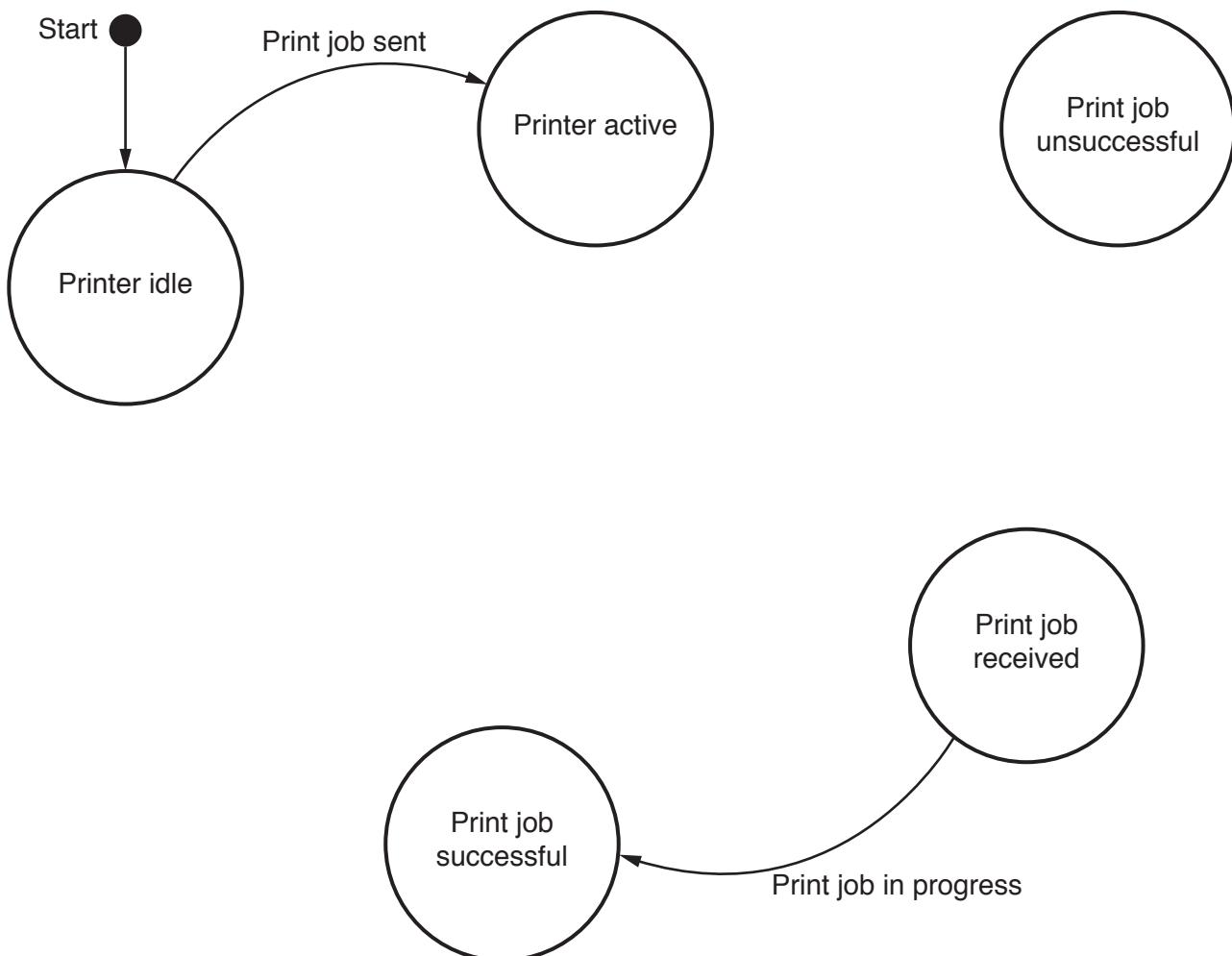
[2]

- (c) The university wants to analyse how a printer and a print server deal with the print jobs.

The following table shows the transitions from one state to another for the process.

Current state	Event	Next state
Printer idle	Print job sent	Printer active
Printer active	Print job added to queue	Print job received
Print job received	Print job in progress	Print job successful
Print job received	Print job in progress	Print job unsuccessful
Print job successful	Check print queue	Printer active
Print job unsuccessful	Error message displayed	Printer active
Printer active	Timeout	Printer idle

Complete the state-transition diagram for the table.



[5]

- (d) The university wants to assess troubleshooting issues with a printer. It wants to use a decision table to do this.

The troubleshooting actions are:

- check the connection from computer to printer, if the error light is flashing **and** the document has not been printed
- check the ink status, if the quality is poor
- check whether there is a paper jam, if the error light is flashing **and** the document has not been printed
- check the paper size selected, if the paper size is incorrect.

- (i) Describe the purpose of a decision table.
-
.....
.....
.....

[2]

- (ii) Complete the rules for the actions in the following decision table.

		Rules							
Conditions	Document printed but the quality is poor	Y	Y	Y	Y	N	N	N	N
	Error light is flashing on printer	Y	Y	N	N	Y	Y	N	N
	Document printed but paper size is incorrect	Y	N	Y	N	Y	N	Y	N
Actions	Check connection from computer to printer								
	Check ink status								
	Check if there is a paper jam								
	Check the paper size selected								

[4]

- (iii) Simplify your solution by removing redundancies.

		Rules						
Conditions	Document printed but the quality is poor							
	Error light is flashing on printer							
	Document printed but paper size is incorrect							
Actions	Check connection from computer to printer							
	Check ink status							
	Check if there is a paper jam							
	Check the paper size selected							

[5]

- (e) There are 1000 students at the university. They will each require a printing account.

Students need to buy printing credits that will be added to their account. Each page printed uses one printing credit.

The university needs software to keep track of the number of printing credits each student has in their account. The university has decided to implement the software using object-oriented programming (OOP).

The following diagram shows the design for the class `PrintAccount`. This includes the attributes and methods.

PrintAccount	
FirstName	: STRING // parameter sent to Constructor()
LastName	: STRING // parameter sent to Constructor()
PrintID	: STRING // parameter sent to Constructor()
Credits	: INTEGER // initialised to 50
Constructor()	// instantiates an object of the PrintAccount class, // and assigns initial values to the attributes
GetName()	// returns FirstName and LastName concatenated // with a space between them
GetPrintID()	// returns PrintID
SetFirstName()	// sets the FirstName for a student
SetLastName()	// sets the LastName for a student
SetPrintID()	// sets the PrintID for a student
AddCredits()	// increases the number of credits for a student
RemoveCredits()	// removes credits from a student account

- (i) Write **program code** for the `Constructor()` method.

Programming language

Program code

[4]

- (ii) Write **program code** for the `SetFirstName()` method.

Programming language

Program code

[2]

[2]

- (iii) Write **program code** for the GetName () method.

Programming language

Program code

[2]

- (iv) The method `AddCredits()` calculates the number of printing credits a student buys and adds the printing credits to the student's account.

- Credits cost \$1 for 25 credits.
 - If a student buys \$20 or more of credits in a single payment, they receive an extra 50 credits.
 - If a student buys between \$10 and \$19 (inclusive) of credits in a single payment, they receive an extra 25 credits.

Payment from a student is stored in the variable MoneyInput. This is passed as a parameter.

Write **program code** for AddCredits(). Use constants for the values that do not change.

Programming language

Program code

[6]

- (v) A global array, `StudentAccounts`, stores 1000 instances of `PrintAccount`.

Write **pseudocode** to declare the array `StudentAccounts`.

[2]

(vi) The main program has a procedure, CreateID(), that:

- takes the first name and last name as parameters
 - creates PrintID that is a concatenation of:
 - the first three letters of the first name in lower case
 - the first three letters of the last name in lower case
 - the character '1'
 - for example, the name Bill Smith would produce "bilsml1"
 - checks if the PrintID created already exists in the global array StudentAccounts:
 - If PrintID does not exist, it creates an instance of PrintAccount in the next free index in StudentAccounts.
 - If PrintID does exist, the number is incremented until a unique ID is created, for example, "bilsml2". It then creates an instance of PrintAccount in the next free index in StudentAccounts.

The global variable `NumberStudents` stores the number of print accounts that have currently been created.

Write program code for the procedure `CreateID()`. Do not write the procedure header.

Programming language

Program code

[8]

[8]

- 2 The following table shows part of the instruction set for a processor, which has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>. <address> can be an absolute address or a symbolic address.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

A programmer writes a program that multiplies two numbers together and outputs the result. The numbers are stored as NUMONE and NUMTWO.

The programmer has started to write the program in the following table. The comment column contains explanations for some of the missing program instructions and data.

Complete the program using the given instruction set.

Label	Op code	Operand	Comment
LOOP:			// load the value from ANSWER
			// add the value from NUMONE
			// load the value from COUNT
			// increment the Accumulator
			// is NUMTWO = COUNT ?
			// if false, jump to LOOP
			// load the value from ANSWER
			// output ANSWER to the screen
			// end of program
NUMONE:	2		
NUMTWO:	4		
COUNT:	0		
ANSWER:	0		

[9]

- 3 Software may not perform as expected. One reason for this is that a syntax error exists in the code.

Identify **three other** reasons why software may not perform as expected.

- 1
-
- 2
-
- 3
-

[3]

- 4 The following table contains definitions related to testing terminology.

Complete the table with the correct testing term to match the definition.

Definition	Term
Software is tested by an in-house team of dedicated testers.
Software is tested by the customer before it is signed off.
Software is tested by a small selection of users before general release.

[3]

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/41

Paper 4 Practical

May/June 2022

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the document **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

A source file is used to answer question 1. The file is called **HighScore.txt**

- 1 The text file HighScore.txt stores the players who have scored the top ten scores in a game, in descending order of score. The file stores the 3-character name of the player, and their integer score, in the order: player, score.

For example, the current top player in the text file:

FYI is the player name

10 000 is the score

The program:

- reads in the data from HighScore.txt
- allows the user to enter a new player name and their score
- if appropriate, inserts the new player (name and score) into the top ten
- writes the top ten players (name and score) into a new text file NewHighScore.txt

- (a) The program stores the players and their scores in an array of 11 elements (10 elements to be read from the file, 1 element to be inserted by the user).

Write a program to declare one or more arrays, as global data structures, to store the player names and their scores.

Save your program as **Question1_J2022**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[2]

- (b) The procedure ReadHighScores () opens the file HighScore.txt and reads the data into the data structure(s) declared in **part 1(a)**.

Write program code to declare the procedure ReadHighScores () .

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[6]

- (c) The procedure `OutputHighScores()` outputs all the values in the data structure(s) in the format:

PlayerName Score

For example, the first two data items: FYI 10000
ABC 9092

Write program code to declare the procedure `OutputHighScores()`.

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[3]

- (d) The main program should first call `ReadHighScores()` and then `OutputHighScores()`.

- (i) Write the program code for the main program.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[2]

- (ii) Test your program.

Take a screenshot to show the output from **part 1(d)(i)**.

Copy and paste the screenshot into **part 1(d)(ii)** in the evidence document.

[1]

- (e) The main program needs to ask the user to input a new player name and a score. If this score is in the top ten then it will create a new top ten list that includes this score.

- (i) Amend the main program to ask the user to input a 3-character player name and an integer score that must be between 1 and 100 000 inclusive.

Save your program.

Copy and paste the program code into **part 1(e)(i)** in the evidence document.

[3]

- (ii) Write program code to declare a procedure that:

- takes the player name and score as parameters
- creates a new top ten list that includes the parameter if appropriate.

Save your program.

Copy and paste the program code into **part 1(e)(ii)** in the evidence document.

[5]

- (iii) Amend the main program to call the procedure from **part 1(e)(ii)**.

Output the contents of the array before inserting the new player name and score, and output the contents of the array after inserting the new player name and score.

Save your program.

Copy and paste the program code into **part 1(e)(iii)** in the evidence document.

[2]

- (iv) Test your program by entering the player name "JKL" and the score "9999".

Take a screenshot to show the output.

Copy and paste the screenshot into **part 1(e)(iv)** in the evidence document.

[1]

- (f) The procedure `WriteTopTen()` stores the new top ten player names and scores in a text file called `NewHighScore.txt`

Write program code to declare the procedure `WriteTopTen()`.

Save your program.

Copy and paste the program code into **part 1(f)** in the evidence document.

[4]

- 2 A computer game is being developed using object-oriented programming.

One element of the game is a balloon. This is designed as the class `Balloon`.

The class has the following attributes and methods.

Balloon	
Health : INTEGER	The health of the balloon
Colour : STRING	The colour of the balloon
DefenceItem : STRING	The item the balloon uses to defend itself
Constructor()	Initialises the defence item and colour using the parameters Initialises health to 100
ChangeHealth()	Takes the change as a parameter and adds this to the health
GetDefenceItem()	Returns the defence item of the object
CheckHealth()	If the health is 0 or less, it returns TRUE, otherwise it returns FALSE

- (a) The constructor takes the name of the defence item and the balloon's colour as parameters and sets these to the attributes. The health is initialised to 100.

Write program code to declare the class `Balloon` and its constructor. Do not write any other methods.

Use your language appropriate constructor.

All attributes should be private. If you are writing in Python include attribute declarations using comments.

Save your program as **Question2_J2022**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[5]

- (b) The get method `GetDefenceItem()` returns the defence item of the object.

Amend your program code to include the get method `GetDefenceItem()`.

Save your program.

Copy and paste the program code into **part 2(b)** in the evidence document.

[2]

- (c) The object's method `ChangeHealth()` takes an integer number as a parameter and adds this to the health attribute of the object.

Amend your program code to include the method `ChangeHealth()`.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[2]

- (d) The object's method `CheckHealth()` returns `TRUE` if the health of the object is 0 or less (no health remaining) and returns `FALSE` otherwise (health remaining).

Amend your program code to include the method `CheckHealth()`.

Save your program.

Copy and paste the program code into **part 2(d)** in the evidence document.

[2]

- (e) Amend the main program to:

- take as input a defence item and colour from the user
- create a new balloon with the identifier `Balloon1` using the data input.

Save your program.

Copy and paste the program code into **part 2(e)** in the evidence document.

[3]

- (f) The function `Defend()`:

- takes a balloon object as a parameter
- takes as input the strength of an opponent from the user
- uses the `ChangeHealth()` method to subtract the strength input from the object's health
- outputs the defence item of the balloon
- checks the health of the object and outputs an appropriate message if it has no health remaining, or if it has health remaining
- returns the amended balloon object.

Write program code to declare the function `Defend()`.

Save your program.

Copy and paste the program code into **part 2(f)** in the evidence document.

[8]

(g) (i) Amend the main program to call the function Defend().

Save your program.

Copy and paste the program code into **part 2(g)(i)** in the evidence document.

[2]

(ii) Test your program using the following inputs:

- balloon defence method "Shield"
- balloon colour "Red"
- strength of opponent 50

Take a screenshot to show the output.

Copy and paste the screenshot into **part 2(g)(ii)** in the evidence document.

[1]

- 3 A program uses a circular queue to store strings. The queue is created as a 1D array, QueueArray, with 10 string items.

The following data is stored about the queue:

- the head pointer initialised to 0
- the tail pointer initialised to 0
- the number of items in the queue initialised to 0.

- (a) Declare the array, head pointer, tail pointer and number of items.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_J2022**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[2]

- (b) The function `Enqueue` is written in pseudocode. The function adds `DataToAdd` to the queue. It returns `FALSE` if the queue is full and returns `TRUE` if the item is added.

The function is incomplete, there are **five** incomplete statements.

```

FUNCTION Enqueue (BYREF QueueArray[] : STRING, BYREF HeadPointer : INTEGER,
                  BYREF TailPointer : INTEGER, NumberItems : INTEGER,
                  DataToAdd : STRING) RETURNS BOOLEAN

IF NumberItems = ..... THEN
    RETURN .....
ENDIF

QueueArray[.....] ← DataToAdd

IF TailPointer >= 9 THEN
    TailPointer ← .....
ELSE
    TailPointer ← TailPointer + 1
ENDIF

NumberItems ← NumberItems .....
RETURN TRUE

ENDFUNCTION

```

Write program code for the function `Enqueue ()`.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[7]

- (c) The function `Dequeue ()` returns "FALSE" if the queue is empty, or it returns the next data item in the queue.

Write program code for the function `Dequeue ()`.

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[6]

(d) (i) Amend the main program to:

- take as input 11 string values from the user
- use the `Enqueue()` function to add each element to the queue
- output an appropriate message to state whether each addition was successful, or not
- call `Dequeue()` function twice and output the return value each time.

Save your program.

Copy and paste the program code into **part 3(d)(i)** in the evidence document.

[5]

(ii) Test your program with the input data:

"A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K"

Take a screenshot to show the output.

Copy and paste the screenshot into **part 3(d)(ii)** in the evidence document.

[1]

BLANK PAGE

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.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

May/June 2017

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **15** printed pages and **1** blank page.

- 1 The following table shows part of the instruction set for a processor which has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
STO	<address>	Store the contents of ACC at the given address.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
CMP	<address>	Compare the contents of ACC with the contents of <address>.
JMP	<address>	Jump to given address.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

- (a) A programmer writes a program that:

- reads a character from the keyboard (assume it will be a capital letter)
- outputs the alphabetical sequence of characters from 'A' to the character input. For example, if the character 'G' is input, the output is:

ABCDEFG

The programmer has started to write the program in the table on the following page. The Comment column contains descriptions for the missing instructions, labels and data.

Complete the following program. Use op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// INPUT character
			// store in CHAR
			// Initialise ACC (ASCII value for 'A' is 65)
			// OUTPUT ACC
			// compare ACC with CHAR
			// if equal jump to end of FOR loop
			// increment ACC
			// jump to LOOP
ENDFOR:	END		
CHAR:			

[8]

(b) The programmer now starts to write a program that:

- tests whether an 8-bit two's complement integer stored at address NUMBER is positive or negative
- outputs 'P' for a positive integer and 'N' for a negative integer.

Complete the following program. Use op codes from the given instruction set.
Show the required value of MASK in binary.

Label	Op code	Operand	Comment
START:			
		MASK	// set to zero all bits except sign bit
			// compare with 0
			// if not equal jump to ELSE
THEN:			// load ACC with 'P' (ASCII value 80)
	JMP	ENDIF	
ELSE:			// load ACC with 'N' (ASCII value 78)
ENDIF:			
	END		
NUMBER:	B00000101		// integer to be tested
MASK:			// value of mask in binary

[7]

- 2 A hash table has these associated operations:

- create hash table
- insert record
- search hash table

A hash table is to be used to store customer records.

Each record consists of a unique customer ID, the record key, and other customer data.

- (a) The following pseudocode declares a customer record structure.

```
TYPE CustomerRecord
    CustomerID : INTEGER
    Data : STRING
ENDTYPE
```

The hash table is to be implemented as a 1D array `Customer` with elements indexed 0 to 199. The procedure to create a hash table will declare and initialise the array by storing 200 records with the `CustomerID` field in each record set to 0.

Complete the **pseudocode**.

```
PROCEDURE CreateHashTable()
```

```
ENDPROCEDURE
```

[4]

- (b) A hashing function `Hash` exists, which takes as a parameter the customer ID and returns an integer in the range 0 to 199 inclusive.

- (i) The procedure, `InsertRecord`, takes as a parameter the customer record to be inserted into the hash table.

The procedure makes use of the function `Hash`. Collisions will be managed using open hashing. This means a collision is resolved by storing the record in the next available location. The procedure will generate an error message if the hash table is full.

Complete the **pseudocode** for the procedure.

```

PROCEDURE InsertRecord(BYVALUE NewCustomer : CustomerRecord)

    TableFull ← FALSE

    // generate hash value

    Index ← .....

    Pointer ← Index    // initialise Pointer variable to hash value

    // find a free table element

    WHILE .....

        Pointer ← .....

        // wrap back to beginning of table if necessary

        IF .....

            THEN
                .....
            ENDIF

        // check if back to original index

        IF .....

            THEN
                TableFull ← TRUE
            ENDIF

        ENDWHILE

        IF .....

            THEN
                .....
            ELSE
                .....
            ENDIF
        ENDIF
    ENDPROCEDURE

```

[9]

- (ii) The function `SearchHashTable` will search for a record in the hash table. The function takes as a parameter the customer ID to be searched for. The function will return the position in the hash table where the record has been saved. If the hash table does not contain the record, the function will return the value `-1`.

You can assume that there is at least one empty record in the hash table.

Complete the **pseudocode** for the function.

```
FUNCTION SearchHashTable(BYVALUE SearchID : INTEGER) RETURNS INTEGER
// generate hash value
Index ← .....
// check each record from index until found or not there
WHILE (.....)
    AND (.....)
    .....
// wrap if necessary
    IF .....  
        THEN  
        .....  
    ENDIF
ENDWHILE
// has customer ID been found?
    IF .....  
        THEN  
        .....  
    ELSE  
        .....  
    ENDIF
ENDFUNCTION
```

[9]

- (iii) A record that is no longer required is deleted.

State the problem that might be caused by this deletion.

[1]

- [1]

- 3 NameList is a 1D array that stores a sorted list of names. A programmer declares the array in pseudocode as follows:

```
NameList : Array[0 : 100] OF STRING
```

The programmer wants to search the list using a binary search algorithm.

The programmer decides to write the search algorithm as a recursive function. The function, Find, takes three parameters:

- Name, the string to be searched for
- Start, the index of the first item in the list to be searched
- Finish, the index of the last item in the list to be searched

The function will return the position of the name in the list, or -1 if the name is not found.

Complete the **pseudocode** for the recursive function.

```
FUNCTION Find(BYVALUE Name : STRING, BYVALUE Start : INTEGER,
              BYVALUE Finish : INTEGER) RETURNS INTEGER

    // base case

    IF ..... THEN
        RETURN -1
    ELSE
        Middle ← .....
        IF ..... THEN
            RETURN .....
        ELSE // general case
            IF SearchItem > .....
                THEN
                    .....
                ELSE
                    .....
            ENDIF
        ENDIF
    ENDIF
ENDFUNCTION
```

4 An ordered linked list Abstract Data Type (ADT) has these associated operations:

- create list
- add item to list
- output list to console

The ADT is to be implemented using object-oriented programming as a linked list of nodes.

Each node consists of data and a pointer.

(a) There are two classes, `LinkedList` and `Node`.

(i) State the term used to describe the relationship between these classes.

..... [1]

(ii) Draw the appropriate diagram to represent this relationship. Do not list the attributes and methods of the classes.

[2]

(b) The design for the Node class consists of:

- attributes
 - Data : STRING
 - Pointer : INTEGER
 - methods
 - CreateNode(Data, Pointer)
 - SetData(Data)
 - SetPointer(Pointer)
 - GetData() RETURNS STRING
 - GetPointer() RETURNS INTEGER

The constructor method sets the attributes to the initial values that are passed as parameters.

Write program code for:

- the `Node` class declaration
 - the constructor.

Programming language used

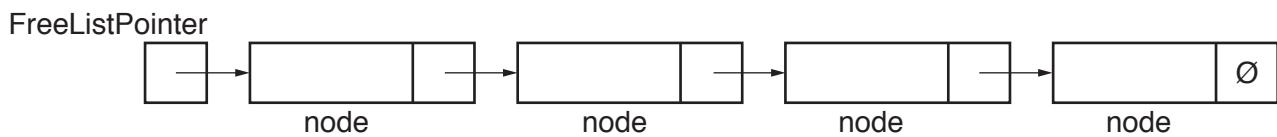
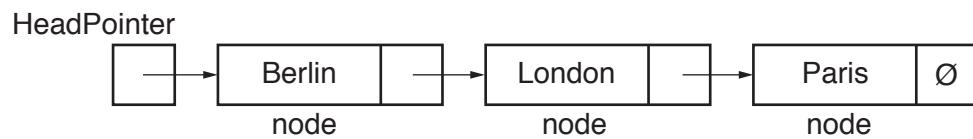
Program code

. [5]

(c) The identifier table for the `LinkedList` class is:

Identifier	Data type	Description
HeadPointer	INTEGER	Pointer to the first node in the ordered list.
FreeListPointer	INTEGER	Pointer to the first node in the free list.
NodeArray	ARRAY [0 : 7] OF Node	1D array stores the nodes that make the ordered linked list. The unused nodes are linked together into a free list.
Constructor()		Constructor instantiates an object of <code>LinkedList</code> class, initialises <code>HeadPointer</code> to be a null pointer and links all nodes to form the free list.
FindInsertionPoint()		Procedure that takes the new data item as the parameter <code>NewData</code> and returns two parameters: <ul style="list-style-type: none"> • <code>PreviousPointer</code>, whose value is: <ul style="list-style-type: none"> ◦ either pointer to node before the insertion point ◦ or the null pointer if the new node is to be inserted at the beginning of the list. • <code>NextPointer</code>, whose value is a pointer to node after the insertion point.
AddToList (NewString)		Procedure that takes as a parameter a unique string and links it into the correct position in the ordered list.
OutputListToConsole()		Procedure to output all the data from the list pointed to by <code>HeadPointer</code> .

The following diagram shows an example of a linked list object. This example list consists of three nodes, linked in alphabetical order of the data strings. The unused nodes are linked to form a free list.



The symbol \emptyset represents a null pointer.

(i) Explain the meaning of the term **null pointer**.

- (ii) Give an appropriate value to represent the null pointer for this design. Justify your answer.

[2]

. [2]

- (iii) Write **program code** for the `LinkedList` class declaration **and** the constructor.

Programming language used

Program code

[7]

- (iv) Write **program code** to instantiate a linked list object with the `contacts` identifier.

Programming language used

Program code

[1]

- (v) The `OutputListToConsole` method is to output all the data stored in the linked list. `HeadPointer` points to the first list node.

Write **program code** for this method.

Programming language used

Program code

. [5]

Question 4 continues on page 14.

(vi) The structured English for the AddToList (NewString) method is as follows:

```

    Make a copy of the value of free list pointer, name it NewNodePointer

    Store new data item in free node pointed to by NewNodePointer

    Adjust free list pointer to point to next free node

    IF linked list is currently empty

        THEN

            Make this node the first node

            Set pointer of this node to null pointer

        ELSE

            Find insertion point using the FindInsertionPoint method

            // FindInsertionPoint provides

            // pointer to previous node and pointer to next node

            IF previous pointer is null pointer

                THEN

                    Link this node to front of list

                ELSE

                    Link this node between previous node and next node
    
```

The FindInsertionPoint method receives the new data item as the parameter NewString. It returns two parameters:

- PreviousPointer, whose value is:
 - either the pointer to the node before the insertion point
 - or the null pointer, if the new node is to be inserted at the beginning of the list.
- NextPointer, whose value is the pointer to the node after the insertion point.

Write program code for the AddToList method.

Programming language used

Program code

[6]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

October/November 2017

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **14** printed pages and **2** blank pages.

- 1 Students are choosing their A Level subjects based on their IGCSE subject results.

A student can take:

- Computer Science, if they have a grade C in Maths or a grade C in Computer Science
- Maths, if they have a grade C in Maths
- Physics, if they have a grade C in Science and a grade C in Maths.

- (a) Complete the decision table.

		Column							
		1	2	3	4	5	6	7	8
Conditions	Grade C in Computer Science	Y	Y	Y	Y	N	N	N	N
	Grade C in Maths	Y	Y	N	N	Y	Y	N	N
	Grade C in Science	Y	N	Y	N	Y	N	Y	N
Actions	Take Computer Science								
	Take Maths								
	Take Physics								

[4]

- (b) Simplify your solution by removing redundancies.

		Column							
		S	T	U	V	W	X	Y	Z
Conditions	Grade C in Computer Science								
	Grade C in Maths								
	Grade C in Science								
Actions	Take Computer Science								
	Take Maths								
	Take Physics								

[3]

- (c) Show how the columns from **part (a)** were simplified to create the columns in **part (b)**.

For example, if columns 5, 6 and 7 were simplified to create column X, then you state this in your answer.

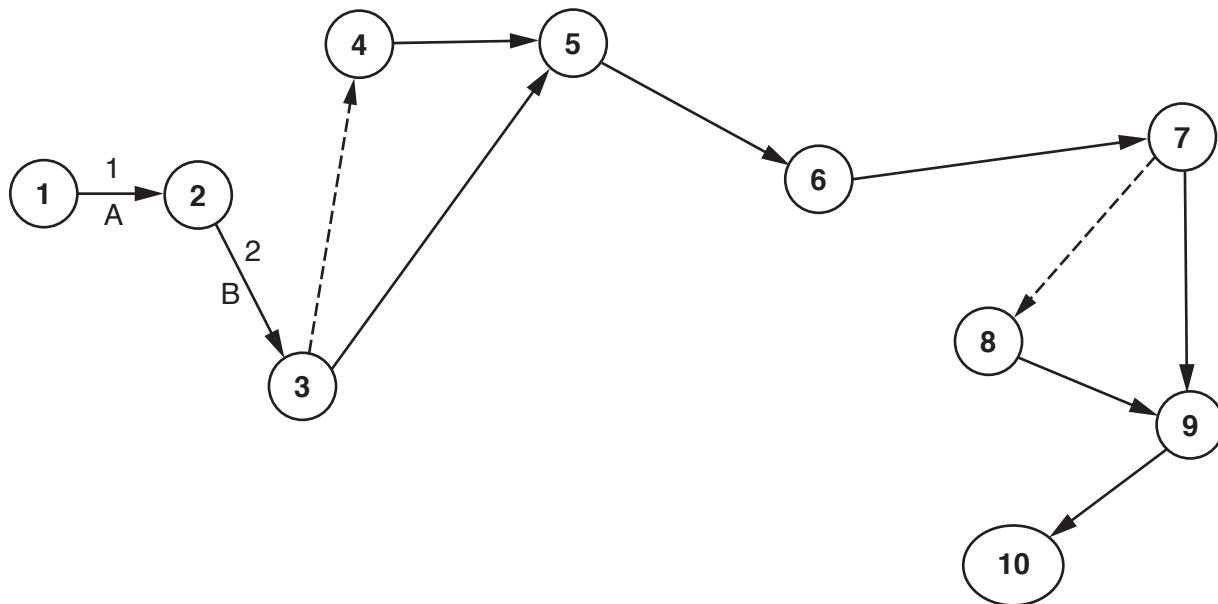
.....
.....
.....
.....
.....
.....

[3]

- 2 (a) A project manager is planning to create a new computer game. The following table shows the activities and the estimated number of weeks to complete each activity.

Activity	Description	Weeks to complete
A	Interview end user	1
B	Produce requirements analysis	2
C	Design program structure	3
D	Design Interface	1
E	Program development	12
F	Black-box testing	2
G	Produce technical documentation	4
H	Acceptance testing	1
I	Installation	1

Complete the labelling of the Program Evaluation Review Technique (PERT) chart using the data in the table. The first two activities have been done for you.



[7]

- (b) State what the dashed lines in the PERT chart represent.

.....
.....

[1]

- 3 A declarative programming language is used to represent the knowledge base:

```

01 room(master_bedroom).
02 room(ensuite_bathroom).
03 room(office).
04 room(spare_bedroom).
05 room(nursery).
06 furniture(bed).
07 furniture(desk).
08 furniture(cot).
09 furniture(wardrobe).
10 furniture(computer).
11 located(bed, master_bedroom).
12 located(bed, spare_bedroom).
13 located(cot, nursery).
14 located(computer, office).
15 located(computer, master_bedroom).

```

These clauses have the following meanings:

Clause	Explanation
01	Master bedroom is a room
06	Bed is an item of furniture
11	Bed is located in the master bedroom

- (a) Corridor is a room that contains a table and a lamp.

Write additional clauses to represent this information.

16

17

18

19

20

[5]

- (b) Using the variable `WhatItem`, the goal:

```
located(WhatItem, master_bedroom).
```

returns:

```
WhatItem = bed, computer
```

Write the result returned by the goal:

```
located(bed, WhichRoom).
```

WhichRoom = [2]

- (c) (i) Clauses to identify rooms that are next to each other need to be stored.

The nursery is next to the master bedroom. This information is stored as:

```
21 nextTo(nursery, master_bedroom).
22 nextTo(master_bedroom, nursery).
```

Explain why both clauses are necessary.

.....
.....
.....
..... [2]

- (ii) The corridor is next to the main bathroom.

Write additional clauses for this fact.

23

24

25

[3]

- (d) B can be moved into A, if B is furniture, A is a room and B is not already in A.

Write this as a rule.

`canBeMovedTo(.....,)`

IF

..... [6]

- 4 (a) The array `Numbers[0 : Max]` stores numbers. An insertion sort can be used to sort these numbers into ascending order.

Complete the following **pseudocode** for the insertion sort algorithm.

```
FOR Pointer ← 1 TO (Max - 1)  
    ItemToInsert ← .....  
    CurrentItem ← .....  
    WHILE (CurrentItem > 0) AND (Numbers[CurrentItem - 1] > ItemToInsert)  
        Numbers[.....] ← Numbers[CurrentItem - 1]  
        CurrentItem ← CurrentItem - 1  
    ENDWHILE  
    Numbers[CurrentItem] ← .....  
ENDFOR
```

[4]

- (b) Identify **two** features of the array `Numbers` that would have an impact on the performance of this insertion sort algorithm.

1
2

[2]

- 5 The following table shows part of the instruction set for a processor. The processor has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

- (a) Six letters are stored, starting at the location labelled LETTERS. A program is needed to perform a linear search on LETTERS to find the letter 'x'. The program counts the number of times 'x' appears in LETTERS.

The following is the pseudocode for the program.

```
FOR COUNT ← 0 TO 5
    IF LETTERS [COUNT] = LETTERTOFIND
        THEN
            FOUND ← FOUND + 1
        ENDIF
    ENDFOR
```

Write this program. Use the op codes from the given instruction set.

Label	Op code	Operand	Comment
START:	LDR	#0	// initialise Index Register
LOOP:			// load LETTERS
			// is LETTERS = LETTERTOFIND ?
			// if not, go to NOTFOUND
			// increment FOUND
NOTFOUND:			// increment COUNT
			// is COUNT = 6 ?
			// if yes, end
			// increment Index Register
			// go back to beginning of loop
ENDP:	END		// end program
LETTERTOFIND:		'x'	
LETTERS:		'd'	
		'u'	
		'p'	
		'l'	
		'e'	
		'x'	
COUNT:		0	
FOUND:		0	

[10]

- (b) Six values are stored, starting at the location VALUES. A program is needed to divide each of the values by 8 and store them back in their original location.

Write this program. Use the op codes from the instruction set on the next page.

Label	Op code	Operand	Comment
START:			// initialise the Index Register
			// load the value from VALUES
			// divide by 8
			// store the new value in VALUES
			// increment the Index Register
			// increment REPS
			// is REPS = 6 ?
			// repeat for next value
	END		
REPS:	0		
VALUES:	22		
	13		
	5		
	46		
	12		
	33		

[10]

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

- 6** A bank has a range of customer accounts, which includes current accounts and savings accounts.

All accounts have:

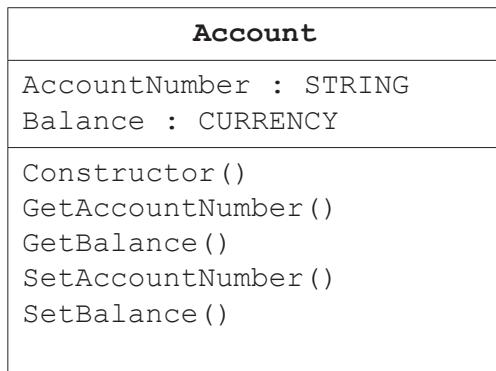
- an account number
 - a balance (amount of money in an account).

A current account has a level (bronze, silver or gold). A monthly fee (\$) is taken from each account.

Savings account customers pay a regular amount (\$) into their account. The payment interval is a number of weeks (for example, 4).

An object-oriented program will be written to process data about the accounts.

(a) Complete the class diagram.



CurrentAccount

SavingsAccount

[3]

- (b)** Write program code to declare the Account class.

Programming language

Program code

..[5]

- (c) Write **program code** to declare the SavingsAccount class. Do not write any get or set methods.

Programming language

Program code

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/42

Paper 4 Practical

May/June 2021

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the document **evidence.doc**.

Make sure that your name, centre number and candidate number will appear on every page of this document. This document will contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

A list is an alternative to an array.

A source file is used to answer question 3. The file is called **TreasureChestData.txt**

- 1 An unordered linked list uses a 1D array to store the data.

Each item in the linked list is of a record type, node, with a field data and a field nextNode.

The current contents of the linked list are:

startPointer	0	Index	data	nextNode
emptyList	5	0	1	1
		1	5	4
		2	6	7
		3	7	-1
		4	2	2
		5	0	6
		6	0	8
		7	56	3
		8	0	9
		9	0	-1

- (a) The following is pseudocode for the record type node.

```

TYPE node
  DECLARE data : INTEGER
  DECLARE nextNode : INTEGER
ENDTYPE

```

Write program code to declare the record type node.

Save your program as **question1**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[2]

- (b) Write program code for the main program.

Declare a 1D array of type `node` with the identifier `linkedList`, and initialise it with the data shown in the table on page 2. Declare the pointers.

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[4]

- (c) The procedure `outputNodes()` takes the array and `startPointer` as parameters. The procedure outputs the data from the linked list by following the `nextNode` values.

- (i) Write program code for the procedure `outputNodes()`.

Save your program.

Copy and paste the program code into **part 1(c)(i)** in the evidence document.

[6]

- (ii) Edit the main program to call the procedure `outputNodes()`.

Take a screenshot to show the output of the procedure `outputNodes()`.

Save your program.

Copy and paste the screenshot into **part 1(c)(ii)** in the evidence document.

[1]

- (d) The function, `addNode()`, takes the linked list and pointers as parameters, then takes as input the data to be added to the end of the `linkedList`.

The function adds the node in the next available space, updates the pointers and returns True. If there are no empty nodes, it returns False.

- (i) Write program code for the function `addNode()`.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[7]

- (ii) Edit the main program to:

- call `addNode()`
- output an appropriate message depending on the result returned from `addNode()`
- call `outputNodes()` twice; once before calling `addNode()` and once after calling `addNode()`.

Save your program.

Copy and paste the program code into **part 1(d)(ii)** in the evidence document.

[3]

- (iii) Test your program by inputting the data value 5 and take a screenshot to show the output.

Save your program.

Copy and paste the screenshot into **part 1(d)(iii)** in the evidence document.

[1]

BLANK PAGE

2 A program stores the following ten integers in a 1D array with the identifier `arrayData`.

10 5 6 7 1 12 13 15 21 8

(a) Write program code for a **new program** to:

- declare the global 1D array, `arrayData`, with ten elements
- initialise `arrayData` in the main program using the data values shown.

Save your program as **question2**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[2]

(b) (i) A function, `linearSearch()`, takes an integer as a parameter and performs a linear search on `arrayData` to find the parameter value. It returns `True` if it was found and `False` if it was not found.

Write program code for the function `linearSearch()`.

Save your program.

Copy and paste the program code into **part 2(b)(i)** in the evidence document.

[6]

(ii) Edit the main program to:

- allow the user to input an integer value
- pass the value to `linearSearch()` as the parameter
- output an appropriate message to tell the user whether the search value was found or not.

Save your program.

Copy and paste the program code into **part 2(b)(ii)** in the evidence document.

[4]

(iii) Test your program with one value that is in the array and one value that is not in the array.

Take a screenshot to show the result of each test.

Save your program.

Copy and paste the screenshots into **part 2(b)(iii)** in the evidence document.

[2]

- (c) The following bubble sort pseudocode algorithm sorts the data in `theArray` into descending numerical order. There are **five** incomplete statements.

```
PROCEDURE bubbleSort()

    DECLARE temp : INTEGER

    FOR x ← 0 to .....
        FOR y ← 0 to .....
            IF theArray[y] ..... theArray[y + 1] THEN
                temp ← theArray[y]
                theArray[y] ← .....
                theArray[y + 1] ← .....
            ENDIF
        NEXT y
    NEXT x
ENDPROCEDURE
```

Write program code for the procedure `bubbleSort()` to sort the data in `arrayData` into descending order.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[6]

- 3 A computer game requires users to travel around a world to find and open treasure chests. Each treasure chest has a mathematics question inside. The user enters the answer. The number of points awarded depends on the number of attempts before the user gives the correct answer.

The program will be created using object-oriented programming (OOP).

The following class diagram describes the class `TreasureChest`.

TreasureChest	
question : STRING answer : INTEGER points : INTEGER	// stores the question // stores the answer // stores the maximum possible number of points available for this chest
constructor()	// takes question, answer and points as parameters and creates an instance of an object
getQuestion()	// returns the question
checkAnswer()	// takes the user's answer as a parameter and returns True if it is correct, otherwise returns False
getPoints()	// takes the number of attempts as a parameter and returns the number of points awarded

- (a) Create a new program.

Write program code to declare the class `TreasureChest`.

Do **not** write any other methods.

The attributes are private.

If you are using the Python programming language, include attribute declarations using comments.

Save your program as **question3**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[5]

- (b) The text file `TreasureChestData.txt` stores data for five questions, in the order of question, answer, points.

For example, the first three lines of the file are for the first question:

```
2*2 question  
4 answer  
10 points
```

Write program code for the procedure, `readData()` to:

- read each question, answer and points from the text file
- create an object of type `TreasureChest` for each question
- declare an array named `arrayTreasure` of type `TreasureChest`
- append each object to the array
- use exception handling to output an appropriate message if the file is not found.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[8]

- (c) The main program repeats each question until the user inputs the correct answer. The number of points awarded depends on the number of attempts before the user gives the correct answer.

- (i) The class `TreasureChest` has a method `getQuestion()` that returns the question.

Write the method `getQuestion()`.

Save your program.

Copy and paste the program code into **part 3(c)(i)** in the evidence document.

[1]

- (ii) The class `TreasureChest` has a method `checkAnswer()` that takes the user's answer as a parameter. It returns `True` if the answer is correct and `False` otherwise.

Write the method `checkAnswer()`.

Save your program.

Copy and paste the program code into **part 3(c)(ii)** in the evidence document.

[3]

- (iii) The class `TreasureChest` has a method `getPoints()` that takes the number of attempts as a parameter.

- If the number of attempts is 1, it returns the value of `points`.
- If the number of attempts is 2, it returns the integer value of `points` divided by 2 (`DIV 2`).
- If the number of attempts is 3 or 4, it returns the integer value of `points` divided by 4 (`DIV 4`).
- If the number of attempts is not 1 or 2 or 3 or 4, it returns 0 (zero).

For example, a question is worth 100 points and the user took 2 attempts to give the correct answer. The user is awarded 50 points ($100 \text{ DIV } 2$).

Write the method `getPoints()`.

Save your program.

Copy and paste the program code into **part 3(c)(iii)** in the evidence document.

[5]

(iv) Write program code for the main program to:

- call the procedure `readData()`
- ask the user to enter a question number between 1 and 5
- output the question that matches the question number entered by the user
- check if the answer input by the user is correct using the method `checkAnswer()`
- repeat the question until the user inputs the correct answer
- count how many times the user attempted the question
- use the method `getPoints()` to return the number of points awarded
- output the number of points the user is awarded.

Save your program.

Copy and paste the program code into **part 3(c)(iv)** in the evidence document.

[7]

(v) Test the program.

Take a screenshot showing the input(s) and output(s) for each of the following two tests.

In the first test:

- select question 1 and answer it correctly the first time.

In the second test:

- select question 5 and answer it correctly the second time.

Save your program.

Copy and paste the screenshots into **part 3(c)(v)** in the evidence document.

[2]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/42

Paper 4 Practical

October/November 2021

2 hours 30 minutes

You will need: Candidate source files (listed on page 2)
evidence.doc



INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the document **evidence.doc**.

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

A source file is used to answer question **2(e)**. The file is called `Pictures.txt`

1 Study the following pseudocode for a recursive function.

```
FUNCTION Unknown (BYVAL X, BYVAL Y : INTEGER) RETURNS INTEGER
    IF X < Y THEN
        OUTPUT X + Y
        RETURN (Unknown(X + 1, Y) * 2)
    ELSE
        IF X = Y THEN
            RETURN 1
        ELSE
            OUTPUT X + Y
            RETURN (Unknown(X - 1, Y) DIV 2)
        ENDIF
    ENDIF
ENDFUNCTION
```

The operator `DIV` returns the integer value after division e.g. `13 DIV 2` would give `6`

(a) Write program code to declare the function `Unknown()`.

Save your program as **question 1**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[3]

- (b) The main program needs to run all **three** of the following function calls and output the result of each call:

Unknown (10, 15)
 Unknown (10, 10)
 Unknown (15, 10)

- (i) For each of the **three** function calls, the main program needs to:

- output the value of the two parameters
- call the function with those parameters
- output the return value.

Write the program code for the main program.

Save your program.

Copy and paste the program code into **part 1(b)(i)** in the evidence document.

[3]

- (ii) Take a screenshot to show the output from **part (b)(i)**.

Copy and paste the screenshot into **part 1(b)(ii)** in the evidence document.

[2]

- (c) Rewrite the function `Unknown ()` as an iterative function, `IterativeUnknown ()`.

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[7]

- (d) The iterative function needs to be called **three** times with the same parameters as in **part (b)**.

- (i) For each of the **three** function calls, the main program needs to:

- output the value of the two parameters
- call the iterative function with those parameters
- output the return value.

Amend the main program to perform these tasks.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[1]

- (ii) Take one or more screenshots to show the output of both functions for each set of parameters.

Copy and paste the screenshot(s) into **part 1(d)(ii)** in the evidence document.

[1]

- 2 A program, written using object-oriented programming, stores pictures as objects.

The program stores the dimensions of the picture (width and height), the colour of the frame (e.g. black), and a description of the picture (e.g. flowers).

The class has the following attributes and methods.

Picture	
Description : STRING Width : INTEGER Height : INTEGER FrameColour : STRING	// stores a description of the picture // stores the width e.g. 30 // stores the height e.g. 40 // stores the colour e.g. black
Constructor() GetDescription() GetHeight() GetWidth() GetColour() SetDescription()	// takes all four values as parameters and sets them to the private attributes // returns the description of the picture // returns the height // returns the width // returns the frame colour // takes the new description as a parameter and writes the value to description

- (a) The constructor takes the picture description, frame colour, height, and width as parameters and sets these to the private attributes.

Write the program code to declare the class `Picture` and its constructor.
Do not write any other methods.

Use your language appropriate constructor. All attributes should be private.

If you are writing in Python programming language, include attribute declarations using comments.

Save your program as **question 2**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[5]

- (b) The four get methods return the associated attribute, for example, `GetDescription()` returns the description of the picture.

Write the **four** get methods.

Save your program.

Copy and paste the program code into **part 2(b)** in the evidence document.

[3]

- (c) The method `SetDescription()` takes a new description as a parameter, and writes this value to the appropriate attribute.

Write the method `SetDescription()`.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[2]

- (d) Write program code to declare an array of type `Picture` with 100 elements.

Save your program.

Copy and paste the program code into **part 2(d)** in the evidence document.

[1]

- (e) The text file `Pictures.txt` stores the data for the pictures in the order: description, width, height, colour.

For example, for the first picture in the text file:

Flowers is the description
45 is the width
50 is the height
black is the frame colour.

The data read into the program from the text file is stored in an array of type `Picture`.
The main program and the function will need to access the array data.

The function `ReadData()`:

- opens the file `Pictures.txt`
- reads the data from the file
- creates a new object of type `Picture` for each picture
- writes each object to the array
- raises an exception if the file cannot be found
- counts and returns the number of pictures in the array.

Write program code for the function `ReadData()`.

Save your program.

Copy and paste the program code into **part 2(e)** in the evidence document.

[8]

- (f) The main program calls the function `ReadData()`.

Write the main program.

Save your program.

Copy and paste the program code into **part 2(f)** in the evidence document.

[2]

- (g) The main program needs to ask the user to input their requirements for a picture. The user will enter the colour of the frame, the maximum width, and the maximum height of the picture.

The program will then search the array of pictures, and output the picture description, the width, and the height of any picture that meets the user's requirements.

The program should allow the user to input the colour in any case (e.g. Silver, silver, or SILVER), and still output the correct results.

Edit the main program to perform the described actions.

Save your program.

Copy and paste the program code into **part 2(g)** in the evidence document.

[7]

- (h) Test your program by inputting the following search criteria:

- BLACK, 100, 100
- silver, 25, 25

Take screenshots to show the output for both search criteria.

Copy and paste the screenshots into **part 2(h)** in the evidence document.

[2]

BLANK PAGE

- 3 An ordered binary tree stores integer data in ascending numerical order.

The data for the binary tree is stored in a 2D array with the following structure:

Index	LeftPointer	Data	RightPointer
[0]	[0]	[1]	[2]
[0]	1	10	2
[1]	-1	5	-1
[2]	-1	16	-1

Each row in the table represents one node on the tree.

The number -1 represents a null pointer.

- (a) The 2D array, `ArrayNodes`, is declared with space for 20 nodes.

Each node has a left pointer, data and right pointer.

The program also initialises the:

- `RootPointer` to -1 (null); this points to the first node in the binary tree
- `FreeNode` to 0; this points to the first empty node in the array.

Write program code to declare `ArrayNodes`, `RootPointer` and `FreeNode` in the main program.

If you are writing in Python programming language, include attribute declarations using comments.

Save your program as **question 3**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[4]

- (b) The procedure `AddNode()` adds a new node to the array `ArrayNodes`.

The procedure needs to:

- take the array, root pointer and free node pointer as parameters
- ask the user to enter the data and read this in
- add the node to the root pointer if the tree is empty
- otherwise, follow the pointers to find the position for the data item to be added
- store the data in the location and update all pointers.

There are **six** incomplete statements in the following pseudocode for the procedure AddNode () .

```

PROCEDURE AddNode (BYREF ArrayNodes[] : ARRAY OF INTEGER,
                  BYREF RootPointer : INTEGER, BYREF FreeNode : INTEGER)
  OUTPUT "Enter the data"
  INPUT NodeData
  IF FreeNode <= 19 THEN
    ArrayNodes[FreeNode, 0] ← -1
    ArrayNodes[FreeNode, 1] ← .....
    ArrayNodes[FreeNode, 2] ← -1
    IF RootPointer = ..... THEN
      RootPointer ← 0
    ELSE
      Placed ← FALSE
      CurrentNode ← RootPointer
      WHILE Placed = FALSE
        IF NodeData < ArrayNodes[CurrentNode, 1] THEN
          IF ArrayNodes[CurrentNode, 0] = -1 THEN
            ArrayNodes[CurrentNode, 0] ← .....
            Placed ← TRUE
          ELSE
            ..... ← ArrayNodes[CurrentNode, 0]
          ENDIF
        ELSE
          IF ArrayNodes[CurrentNode, 2] = -1 THEN
            ArrayNodes[CurrentNode, 2] ← FreeNode
            Placed ← .....
          ELSE
            CurrentNode ← ArrayNodes[CurrentNode, 2]
          ENDIF
        ENDIF
      ENDWHILE
    ENDIF
    FreeNode ← ..... + 1
  ELSE
    OUTPUT "Tree is full"
  ENDIF
ENDPROCEDURE

```

Write **program code** for the procedure AddNode () .

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[8]

- (c) The procedure `PrintAll()` outputs the data in each element in `ArrayNodes`, in the order they are stored in the array.

Each element is printed in a row in the order:

LeftPointer Data RightPointer

For example:

1	20	-1
-1	10	-1

Write program code for the procedure `PrintAll()`.

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[4]

- (d) The main program should loop 10 times, each time calling the procedure `AddNode()`. It should then call the procedure `PrintAll()`.

(i) Edit the main program to perform the actions described.

Save your program.

Copy and paste the program code into **part 3(d)(i)** in the evidence document.

[3]

(ii) Test the program by entering the data:

10
5
15
8
12
6
20
11
9
4

Take a screenshot to show the output after the given data are entered.

Copy and paste the screenshot into **part 3(d)(ii)** in the evidence document.

[1]

- (e) An in-order tree traversal visits the left node, then the root (and outputs this), then visits the right node.
- (i) Write a recursive procedure, `InOrder()`, to perform an in-order traversal on the tree held in `ArrayNodes`.

Save your program.

Copy and paste the program code into **part 3(e)(i)** in the evidence document.

[7]

- (ii) Test the procedure `InOrder()` with the same data entered in **part (d)(ii)**.

Take a screenshot to show the output after entering the data.

Copy and paste the screenshot into **part 3(e)(ii)** in the evidence document.

[1]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

May/June 2019

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **18** printed pages and **2** blank pages.

- 1 (a) A stack contains the values 'red', 'blue', 'green' and 'yellow'.



- (i) Show the contents of the stack in **part(a)** after the following operations.

POP()

PUSH('purple')

PUSH('orange')



[1]

- (ii) Show the contents of the stack from **part(a)(i)** after these further operations.

POP ()

POP ()

PUSH ('brown')

POP ()

PUSH ('black')



[1]

- (b) A queue is an alternative Abstract Data Type (ADT).

Describe a **queue**.

.....
.....
.....
.....
.....
.....
.....
.....

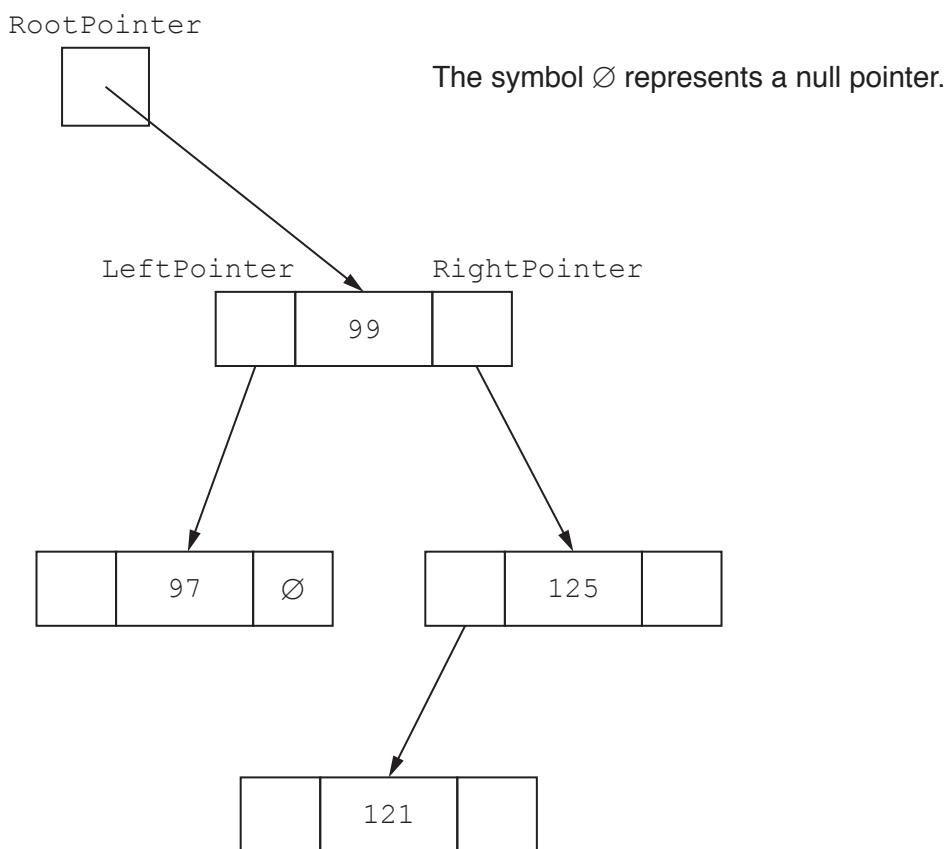
[3]

- 2 A computer games club wants to run a competition. The club needs a system to store the scores achieved in the competition.

A selection of score data is as follows:

99, 125, 121, 97, 109, 95, 135, 149

- (a) A linked list of nodes will be used to store the data. Each node consists of the data, a left pointer and a right pointer. The linked list will be organised as a binary tree.
- (i) Complete the binary tree to show how the score data above will be organised.



[5]

- (ii) The following diagram shows a 2D array that stores the nodes of the binary tree's linked list.

Add the correct pointer values to complete the diagram, using your answer from part (a)(i).

RootPointer	Index	LeftPointer	Data	RightPointer
<input type="text"/>	0		99	
	1		125	
	2		121	
FreePointer	3		97	
<input type="text"/>	4		109	
	5		95	
	6		135	
	7		149	
	8			

[6]

- (b) The club also considers storing the data in the order in which it receives the scores as a linked list in a 1D array of records.

The following pseudocode algorithm searches for an element in the linked list.

Complete the **six** missing sections in the algorithm.

```
FUNCTION FindElement(Item : INTEGER) RETURNS .....  
..... ← RootPointer  
WHILE CurrentPointer ..... NullPointer  
IF List[CurrentPointer].Data <> .....  
THEN  
    CurrentPointer ← List[.....] . Pointer  
ELSE  
    RETURN CurrentPointer  
ENDIF  
ENDWHILE  
CurrentPointer ← NullPointer  
..... CurrentPointer  
ENDFUNCTION
```

[6]

- (c) The games club is looking at two programming paradigms: imperative and object-oriented programming paradigms.

Describe what is meant by the **imperative programming paradigm** and the **object-oriented programming paradigm**.

(i) Imperative

.....
.....
.....
.....
..... [3]

(ii) Object-oriented

.....
.....
.....
.....
..... [3]

- (d) Players complete one game to place them into a category for the competition. The games club wants to implement a program to place players into the correct category. The programmer has decided to use object-oriented programming (OOP).

The highest score that can be achieved in the game is 150. Any score less than 50 will not qualify for the competition. Players will be placed in a category based on their score.

The following diagram shows the design for the class `Player`. This includes the properties and methods.

Player	
<code>Score : INTEGER</code>	// initialised to 0
<code>Category : STRING</code>	// "Beginner", "Intermediate", // "Advanced" or "Not Qualified", initialised // to "Not Qualified"
<code>PlayerID : STRING</code>	// initialised with the parameter InputPlayerID
<code>Create()</code>	// method to create and initialise an object using // language-appropriate constructor
<code>SetScore()</code>	// checks that the Score parameter has a valid value // if so, assigns it to Score
<code>SetCategory()</code>	// sets Category based on player's Score
<code>SetPlayerID()</code>	// allows a player to change their PlayerID // validates the new PlayerID
<code>GetScore()</code>	// returns Score
<code>GetCategory()</code>	// returns Category
<code>GetPlayerID()</code>	// returns PlayerID

- (i) The constructor receives the parameter `InputPlayerID` to create the `PlayerID`. Other properties are initialised as instructed in the class diagram.

Write program code for the Create() constructor method.

Programming language

Program code

[5]

[5]

- (ii) Write **program code** for the following **three** get methods.

Programming language

GetScore ()

Program code

.....
.....
.....
.....

GetCategory ()

Program code

.....
.....
.....
.....

GetPlayerID ()

Program code

.....
.....
.....
.....

[4]

- (iii) The method `SetPlayerID()` asks the user to input the new player ID and reads in this value.

It checks that the length of the `PlayerID` is less than or equal to 15 characters and greater than or equal to 4 characters. If the input is valid, it sets this as the `PlayerID`, otherwise it loops until the player inputs a valid `PlayerID`.

Use suitable input and output messages.

Write program code for SetPlayerID().

Programming language

Program code

[4]

[4]

- (iv) The method `SetScore()` checks that its `INTEGER` parameter `ScoreInput` is valid. If it is valid, it is then set as `Score`. A valid `ScoreInput` is greater than or equal to 0 and less than or equal to 150.

If the ScoreInput is valid, the method sets Score and returns TRUE.

If the `ScoreInput` is not valid, the method does not set `Score`, displays an error message, and it returns `FALSE`.

Write program code for SetScore (ScoreInput : INTEGER).

Programming language

Program code

[5]

[5]

- (v) Write **program code** for the method `SetCategory()`. Use the properties and methods in the original class definition.

Players will be placed in one of the following categories.

Category	Criteria
Advanced	Score is greater than 120
Intermediate	Score is greater than 80 and less than or equal to 120
Beginner	Score is greater than or equal to 50 and less than or equal to 80
Not Qualified	Score is less than 50

Programming language

Program code

[4]

- (vi) Joanne has played the first game to place her in a category for the competition.

The procedure `CreatePlayer()` performs the following tasks.

- allows the player ID and score to be input with suitable prompts
 - creates an instance of `Player` with the identifier `JoannePlayer`
 - sets the score for the object
 - sets the category for the object
 - outputs the category for the object

Write program code for the CreatePlayer() procedure.

Programming language

Program code

[8]

- (e) The programmer wants to test that the correct category is set for a player's score.

As stated in **part (d)(v)**, players will be placed in one of the following categories.

Category	Criteria
Advanced	Score is greater than 120
Intermediate	Score is greater than 80 and less than or equal to 120
Beginner	Score is greater than or equal to 50 and less than or equal to 80
Not Qualified	Score is less than 50

Complete the table to provide test data for each category.

Category	Type of test data	Example test data
Beginner	Normal	
	Abnormal	
	Boundary	
Intermediate	Normal	
	Abnormal	
	Boundary	
Advanced	Normal	
	Abnormal	
	Boundary	

[3]

- (f) In part (b), the club stored scores in a 1D array. This allows the club to sort the scores.

The following is a sorting algorithm in pseudocode.

```
NumberOfScores ← 5  
  
FOR Item ← 1 TO NumberOfScores - 1  
  
    InsertScore ← ArrayData[Item]  
  
    Index ← Item - 1  
  
    WHILE (ArrayData[Index] > InsertScore) AND (Index >= 0)  
        ArrayData[Index + 1] ← ArrayData[Index]  
  
        Index ← Index - 1  
  
    ENDWHILE  
  
    ArrayData[Index + 1] ← InsertScore  
  
ENDFOR
```

- (i) Give the name of this algorithm.

..... [1]

- (ii) State the name of **one** other sorting algorithm.

..... [1]

- (iii) Complete a dry run of the algorithm using the following trace table.

[7]

- 3 Some algorithms can be written using recursion.

- (a) State **two** features of recursion.

Feature 1

Feature 2

[2]

- (b) Explain what a compiler has to do to implement recursion.

.....
.....
.....
.....
.....
..... [3]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/42

Paper 4 Practical

October/November 2023

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **16** pages. Any blank pages are indicated.

Open the evidence document, **evidence.doc**

Make sure that your name, centre number and candidate number appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: **evidence_zz999_9999**

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

One source file is used to answer **Question 1**. The file is called **StackData.txt**

1 A program stores lower-case letters in two stacks.

One stack stores vowels (a, e, i, o, u) and one stack stores consonants (letters that are not vowels).

Each stack is implemented as a 1D array.

(a) (i) Write program code to declare two 1D global arrays: **StackVowel** and **StackConsonant**.

Each array needs to store up to 100 letters. The index of the first element in each array is 0.

If you are writing in Python, include declarations using comments.

Save your program as **Question1_N23**.

Copy and paste the program code into part **1(a)(i)** in the evidence document.

[2]

(ii) The global variable **VowelTop** is a pointer that stores the index of the next free space in **StackVowel**.

The global variable **ConsonantTop** is a pointer that stores the index of the next free space in **StackConsonant**.

VowelTop and **ConsonantTop** are both initialised to 0.

Write program code to declare and initialise the two variables.

If you are writing in Python, include declarations using comments.

Save your program.

Copy and paste the program code into part **1(a)(ii)** in the evidence document.

[1]

(b) (i) The procedure `PushData()` takes one letter as a parameter.

If the parameter is a vowel, it is pushed onto `StackVowel` and the relevant pointer updated.

If the stack is full, a suitable message is output.

If the parameter is a consonant, it is pushed onto `StackConsonant` and the relevant pointer updated.

If the stack is full, a suitable message is output.

You do **not** need to validate that the parameter is a letter.

Write program code for `PushData()`.

Save your program.

Copy and paste the program code into part 1(b)(i) in the evidence document.

[6]

(ii) The file `StackData.txt` stores 100 lower-case letters.

The procedure `ReadData()` reads each letter from the file and uses `PushData()` to push each letter onto its appropriate stack.

Use appropriate exception handling if the file does not exist.

Write program code for `ReadData()`.

Save your program.

Copy and paste the program code into part 1(b)(ii) in the evidence document.

[6]

(c) The function `PopVowel()` removes and returns the data at the top of `StackVowel` and updates the relevant pointer(s).

The function `PopConsonant()` removes and returns the data from the top of `StackConsonant` and updates the relevant pointer(s).

If either stack is empty, the string "No data" must be returned.

Write program code to declare `PopVowel()` and `PopConsonant()`.

Save your program.

Copy and paste the program code into part 1(c) in the evidence document.

[5]

(d) The program first needs to call `ReadData()` and then:

1. prompt the user to input their choice of vowel or consonant
2. take, as input, the user's choice
3. depending on the user's choice, call `PopVowel()` or `PopConsonant()` and store the return value.

The three steps are repeated until 5 letters have been successfully returned and stored.

If either stack is empty at any stage, an appropriate message must be output.

Once 5 letters have been successfully returned and stored, they are output on one line, for example:

abyti

(i) Write program code for the main program.

Save your program.

Copy and paste the program code into part 1(d)(i) in the evidence document.

[6]

(ii) Test your program with the following inputs:

vowel

consonant

consonant

vowel

vowel

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part 1(d)(ii) in the evidence document.

[1]

BLANK PAGE

- 2** An integer is said to be divisible by another integer if the result of the division is also an integer.

For example:

10 is divisible by 1, 2, 5 and 10:

- $10 \div 1 = 10$
- $10 \div 2 = 5$
- $10 \div 5 = 2$
- $10 \div 10 = 1$

10 is not divisible by 4:

- $10 \div 4 = 2.5$

1, 2, 5 and 10 are said to be the divisors of 10.

The iterative function `IterativeCalculate()` totals all the divisors of its integer parameter and returns this total.

Example 1: if the parameter is 10, the total will be 18 ($1 + 2 + 5 + 10$) .

Example 2: if the parameter is 4, the total will be 7 ($1 + 2 + 4$) .

A pseudocode algorithm for `IterativeCalculate()` is shown.

```

FUNCTION IterativeCalculate(Number : INTEGER) RETURNS INTEGER

    DECLARE Total : Integer
    DECLARE ToFind : Integer
    ToFind ← Number
    Total ← 0

    WHILE Number <> 0
        IF ToFind MODULUS Number = 0 THEN
            Total ← Total + Number
        ENDIF
        Number ← Number - 1
    ENDWHILE
    RETURN Total
ENDFUNCTION

```

The operator `MODULUS` calculates the remainder when one number is divided by another.

- (a) (i) Write program code for `IterativeCalculate()`.

Save your program as **Question2_N23**.

Copy and paste the program code into part **2(a)(i)** in the evidence document.

[5]

- (ii) Write program code to call `IterativeCalculate()` with 10 as the parameter and output the return value.

Save your program.

Copy and paste the program code into part **2(a)(ii)** in the evidence document.

[2]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **2(a)(iii)** in the evidence document.

[1]

- (b) `IterativeCalculate()` has been rewritten as the recursive function `RecursiveValue()`.

A pseudocode algorithm for `RecursiveValue()` is given. The function is incomplete.

```

FUNCTION RecursiveValue(Number : INTEGER, ToFind : INTEGER)
    RETURNS INTEGER

    IF Number = ..... THEN
        RETURN 0
    ELSE
        IF ToFind ..... Number = 0 THEN
            RETURN ..... + RecursiveValue(Number - 1, ToFind)
        ELSE
            ..... RecursiveValue(Number - 1, ....)
        ENDIF
    ENDIF
ENDFUNCTION

```

- (i) Write program code for `RecursiveValue()`.

Save your program.

Copy and paste the program code into part **2(b)(i)** in the evidence document.

[7]

- (ii) Write program code to call `RecursiveValue()` with 50 as the first parameter and 50 as the second parameter and output the return value.

Save your program.

Copy and paste the program code into part **2(b)(ii)** in the evidence document.

[1]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **2(b)(iii)** in the evidence document.

[1]

BLANK PAGE

- 3 A computer game is written using object-oriented programming.

The game has multiple characters.

The class `Character` stores data about the game characters. Each character has a name, date of birth, intelligence value and speed value.

Character	
<code>CharacterName : STRING</code>	stores the name of the character
<code>DateOfBirth : DATE</code>	stores the date of birth of the character
<code>Intelligence : REAL</code>	stores the intelligence value of the character
<code>Speed : INTEGER</code>	stores the speed value of the character
<code>Constructor()</code>	initialises <code>CharacterName</code> , <code>DateOfBirth</code> , <code>Intelligence</code> and <code>Speed</code> to the parameter values
<code>SetIntelligence()</code>	assigns the value of the parameter to <code>Intelligence</code>
<code>GetIntelligence()</code>	returns the value of <code>Intelligence</code>
<code>GetName()</code>	returns the name of the character
<code>ReturnAge()</code>	calculates and returns the age of the character as an integer
<code>Learn()</code>	increases the value of <code>Intelligence</code> by 10%

- (a) (i) Write program code to declare the class `Character` and its constructor.

Do **not** declare the other methods.

Use your programming language's appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_N23**.

Copy and paste the program code into part 3(a)(i) in the evidence document.

[5]

- (ii) The get methods `GetIntelligence()` and `GetName()` return the attribute values.

Write program code for the methods `GetIntelligence()` and `GetName()`.

Save your program.

Copy and paste the program code into part 3(a)(ii) in the evidence document.

[3]

- (iii) The method `SetIntelligence()` assigns the value of its parameter to the attribute.

Write program code for `SetIntelligence()`.

Save your program.

Copy and paste the program code into part 3(a)(iii) in the evidence document.

[2]

- (iv) The method `Learn()` increases the current value of `Intelligence` by 10%.

Write program code for `Learn()`.

Save your program.

Copy and paste the program code into part 3(a)(iv) in the evidence document.

[1]

- (v) The method `ReturnAge()` calculates and returns the age of the character in years as an integer.

Assume that the current year is 2023 and **only** use the year from the date of birth for the calculation. For example, the method returns 18 if the character was born on any date in 2005.

Write program code for the method `ReturnAge()`.

Save your program.

Copy and paste the program code into part 3(a)(v) in the evidence document.

[2]

- (b) (i) Write program code to create a new instance of `Character` with the identifier `FirstCharacter`.

The name of the character is Royal, date of birth is 1 January 2019, intelligence is 70 and speed is 30.

Save your program.

Copy and paste the program code into part 3(b)(i) in the evidence document.

[2]

- (ii) Write program code to call the method `Learn()` for the character created in part 3(b)(i).

Output the name, age and intelligence of the character in an appropriate message.

Save your program.

Copy and paste the program code into part 3(b)(ii) in the evidence document.

[3]

(iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part 3(b)(iii) in the evidence document.

[1]

- (c) The class `MagicCharacter` inherits from the class `Character`. A magic character has an element, for example, water. This element changes how they learn. The magic character's element is stored in the additional attribute `Element`.

MagicCharacter	
Element : STRING	stores the element for the character
Constructor()	takes <code>Element</code> , <code>CharacterName</code> , <code>DateOfBirth</code> , <code>Intelligence</code> and <code>Speed</code> as parameters calls its parent class constructor with the appropriate values initialises <code>Element</code> to its parameter value
Learn()	alters the intelligence of the character depending on the character's element

- (i) Write program code to declare the class `MagicCharacter` and its constructor.

Do **not** declare the other method.

Use your programming language's appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into part 3(c)(i) in the evidence document.

[5]

- (ii) The method `Learn()` overrides the parent class method and increases the intelligence depending on the character's element.

- If the element is fire or water, intelligence increases by 20%.
- If the element is earth, intelligence increases by 30%.
- If the element is not fire, water or earth the intelligence increases by 10%.

Write program code for `Learn()`.

Save your program.

Copy and paste the program code into part 3(c)(ii) in the evidence document.

[3]

- (d) (i) Write program code to create a new instance of `MagicCharacter` with the identifier `FirstMagic`.

The name of the character is Light, date of birth is 3 March 2018, intelligence is 75, speed is 22 and element is fire.

Save your program.

Copy and paste the program code into part 3(d)(i) in the evidence document.

[2]

- (ii) Write program code to call the method `Learn()` for the character created in part 3(d)(i).

Output the name, age and intelligence of the character in an appropriate message.

Save your program.

Copy and paste the program code into part 3(d)(ii) in the evidence document.

[1]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part 3(d)(iii) in the evidence document.

[1]

BLANK PAGE

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.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

October/November 2017

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

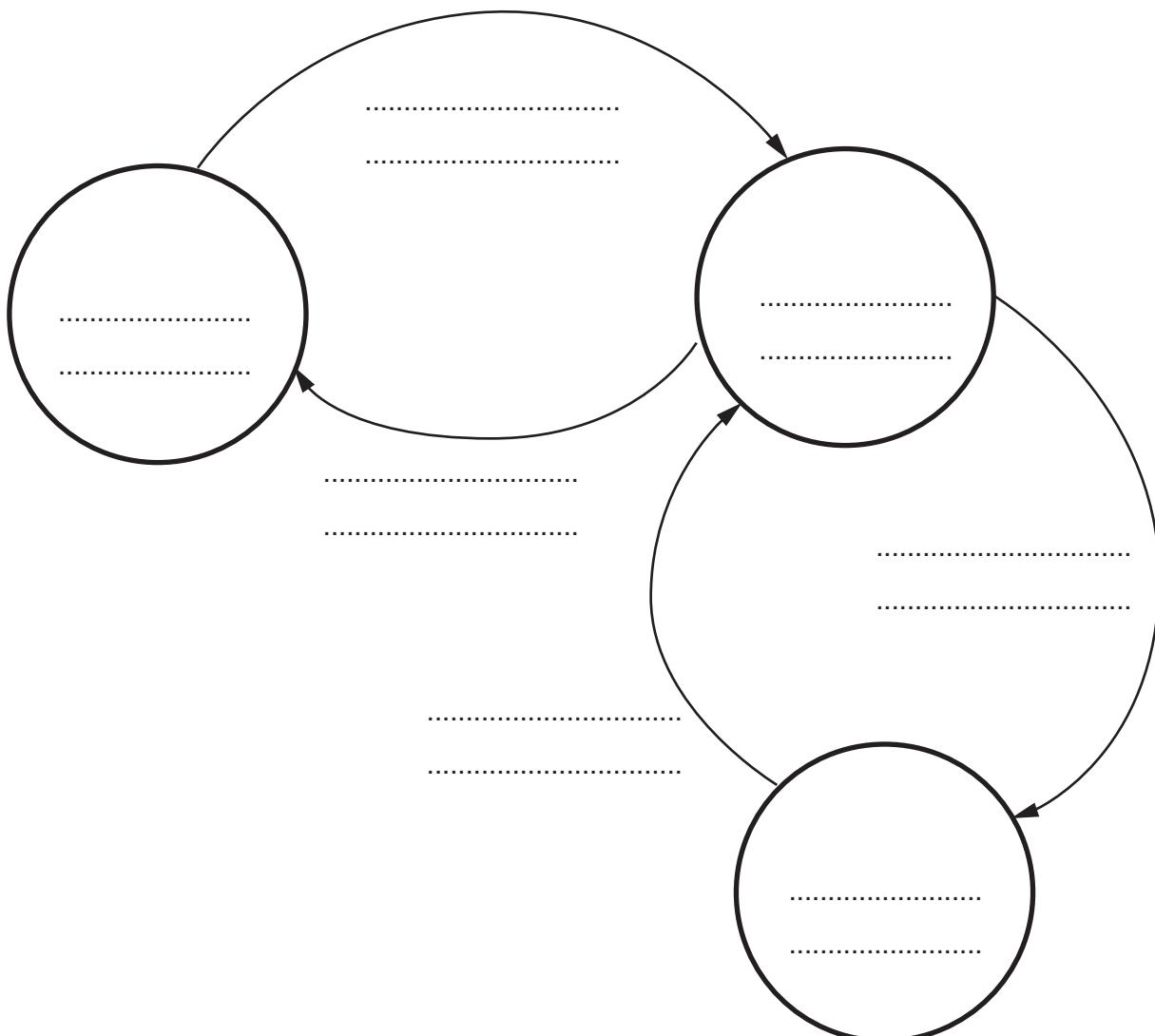
- 1 A greenhouse has a window that automatically opens and closes depending on the internal temperature.

If the temperature rises above 20°C , the window half opens. If the temperature rises above 30°C , the window fully opens. If the temperature drops below 25°C , the window returns to being half open. If the temperature drops below 15°C , the window fully closes.

The window has three possible states: **Closed**, **Half Open** and **Fully Open**.

Current state	Event	Next state
Closed	Temperature rises above 20°C	Half Open
Half Open	Temperature drops below 15°C	Closed
Half Open	Temperature rises above 30°C	Fully Open
Fully Open	Temperature drops below 25°C	Half Open

Complete the state-transition diagram for the window:



[7]

- 2 (a) (i) State how repetition is shown in a Jackson Structured Programming (JSP) structure diagram.

.....
..... [1]

- (ii) State how selection is shown in a JSP structure diagram.

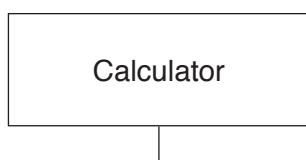
.....
..... [1]

- (b) A simple calculator is to be created.

The calculator is to be used as follows:

- User inputs 2 numbers (x and y).
- User inputs an operator (+, -, * or /).
- The calculator computes the answer.
- The calculator displays the answer.

Draw a JSP diagram for the calculator. The first element is provided.



[5]

- 3 A declarative programming language is used to represent the following knowledge base:

```

01 person(jane).
02 person(ahmed).
03 person(caroline).
04 person(stuart).
05 food(chocolate).
06 food(sushi).
07 food(pizza).
08 food(chilli).
09 likes(jane, pizza).
10 likes(ahmed, chocolate).
11 likes(ahmed, pizza).
12 likes(jane, chilli).
13 likes(stuart, sushi).
14 dislikes(stuart, chocolate).
15 dislikes(jane, sushi).
16 dislikes(caroline, pizza).

```

These clauses have the following meanings:

Clause	Explanation
01	Jane is a person
05	Chocolate is a food
09	Jane likes pizza
14	Stuart dislikes (does not like) chocolate

- (a) Mimi is a person who likes chocolate but does not like sushi or lettuce.

Write additional clauses to represent this information.

- 17
- 18
- 19
- 20
- 21

[5]

- (b) Using the variable PersonName, the goal:

likes(PersonName, pizza).

returns:

PersonName = jane, ahmed.

Write the result that is returned by the goal:

likes(ahmed, FoodItem).

FoodItem = [2]

- (c) B might like A, if B is a person, A is a food and B does not dislike A.

Write this as a rule.

might_like(.....,)

IF
.....
..... [6]

- 4 The following table shows part of the instruction set for a processor. The processor has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

- (a) A program stores a letter. The user is allowed nine attempts to guess the stored letter. The program outputs "?" and the user guesses a letter. If the user guesses the letter, the program outputs "*".

The following is pseudocode for this program.

```

REPEAT
    OUTPUT '?'
    INPUT GUESS
    IF GUESS = LETTERTOGUESS
        THEN
            OUTPUT '*'
            BREAK
    ELSE
        ATTEMPTS ← ATTEMPTS + 1
    ENDIF
UNTIL ATTEMPTS = 9

```

Write this program. Use the op codes from the instruction set provided.

Label	Op code	Operand	Comment
START:	LDM	#63	// load ASCII value for '?'
			// OUTPUT '?'
			// input GUESS
			// compare with stored letter
			// if correct guess, go to GUESSED
			// increment ATTEMPTS
			// is ATTEMPTS = 9 ?
			// if out of guesses, go to ENDP
			// go back to beginning of loop
GUESSED:	LDM	#42	// load ASCII for '*'
			// OUTPUT '*'
ENDP:	END		// end program
ATTEMPTS:		0	
LETTERTOGUESS:		'a'	

[11]

- (b) Five numbers are stored, starting in the location labelled NUMBERS. A program is needed to multiply each of the numbers by 4 and store them back in their original location.

Write this program. Use the op codes from the instruction set on the opposite page.

Label	Op code	Operand	Comment
START:			// initialise the Index Register
			// load the value from NUMBERS
			// multiply by 4
			// store the new value in NUMBERS
			// increment the Index Register
			// increment COUNT
			// is COUNT = 5 ?
			// repeat for next number
ENDP:	END		
COUNT:	0		
NUMBERS:	22		
	13		
	5		
	46		
	12		

[10]

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

5 Large development projects require careful resource management.

- (a) (i) Name an appropriate project management tool that helps the manager to work out the estimated length of time it takes for the project to complete.

.....
.....

[1]

- (ii) Explain how, during the planning stage of the project, the manager would use the tool you named in part (a)(i).

.....
.....
.....
.....
.....

[3]

- (b) (i) Different programmers have been writing independent modules. The modules now need to be combined to create the final system.

Name the type of testing required at this stage.

.....
.....

[1]

- (ii) Name the final testing stage required before the system becomes operational.

.....
.....

[1]

- 6 A programmer wants to create a computer simulation of animals searching for food in a desert. The desert is represented by a 40 by 40 grid. Each position in the grid is represented by a pair of coordinates. 'A' represents an animal and 'F' represents food. At the start of the simulation, the grid contains 5 animals and 1 food source.

The following is an example of part of the grid.

	0	1	2	3	4	...	37	38	39
0	A					..			
1			F			..			
2						..		A	
3				A		..			
..
38				A		..	A		
39						..			

A timer is used. In each time interval, each animal randomly moves 0 or 1 position in a random direction. The program generates this movement by computing two random numbers, each of which can be -1, 0 or 1. The program adds the first random number to the across number and the second random number to the down number representing the animal's position.

For example:

- if 0 and 1 are generated, the across value does not change, the down value increases by 1
- if -1 and 1 are generated, the across value decreases by 1, and the down value increases by 1.

Each animal has an individual score. If the animal moves to a position in the grid with food ('F'):

- the animal's score increases by 1
- the food disappears
- one new animal ('A') is randomly generated and added to the grid (to a maximum of 20 animals)
- one new food ('F') is randomly generated and added to the grid.

The simulation is to be implemented using object-oriented programming.

The programmer has designed two classes, Desert and Animal.

The Desert class consists of:

- attributes
 - Grid
 - StepCounter
 - AnimalList
 - NumberOfAnimals
- methods
 - Constructor
 - IncrementStepCounter
 - GenerateFood
 - DisplayGrid

Each attribute consists of a value and a get and set method that allow access to the attributes.

The following table describes the attributes and methods for the Animal class.

Identifier	Data type	Description
Constructor()		<p>Instantiate an object of the Animal class</p> <ul style="list-style-type: none"> Generate a pair of random numbers between 0 and 39. Place animal at that random position. Initialise the animal's score to 0.
EatFood()		<ul style="list-style-type: none"> Delete the food. Increase the score of the animal that called the method. Call the GenerateFood method of the Desert class. Call the Constructor method of the Animal class.
Move()		<ul style="list-style-type: none"> Call the GenerateChangeInCoordinate method for each coordinate (across or down number) of the animal's position. Moves the animal to the new space. If there is food in the new position, call the EatFood method.
Score	INTEGER	Initialised to 0
Across	INTEGER	The across value, between 0 and 39
Down	INTEGER	The down value, between 0 and 39

- (a) Write **program code** to declare the attributes and constructor for the `Animal` class.

You only need to write the set and get methods for the attribute Across.

You should also write:

- the constructor for the class
 - set and get methods for the `Across` attribute only.

Programming language

Program code

.[6]

(b) The Constructor method of the Desert class:

- initialises an empty grid
 - creates 5 animal objects which are added to the `AnimalList` (an array of animal objects currently on the grid)
 - generates one food
 - sets the `StepCounter` to 0.

Write program code for the Constructor method.

Programming language

Program code

.[5]

(c) (i) The function GenerateChangeInCoordinate:

- receives a coordinate (across or down number) as a parameter
 - checks whether the coordinate's value is at a boundary of the grid
 - returns a random change (-1 , 0 or 1) that will keep the animal's position within the grid.

Write program code for the `GenerateChangeInCoordinate` function.

Programming language

Program code

[4]

.[4]

- (ii) The Move method uses the GenerateChangeInCoordinate function to calculate the new Across and Down values for an animal. If there is food in the new position in the grid, the animal eats the food.

Write program code for the Move method.

Programming language

Program code

[4]

. [4]

- (d) The programmer plans to add a graphic display to the program. The programmer will make use of a program library.

Explain what is meant by a program library.

[2]

.[2]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

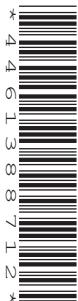
CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

October/November 2019

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of 17 printed pages and 3 blank pages.

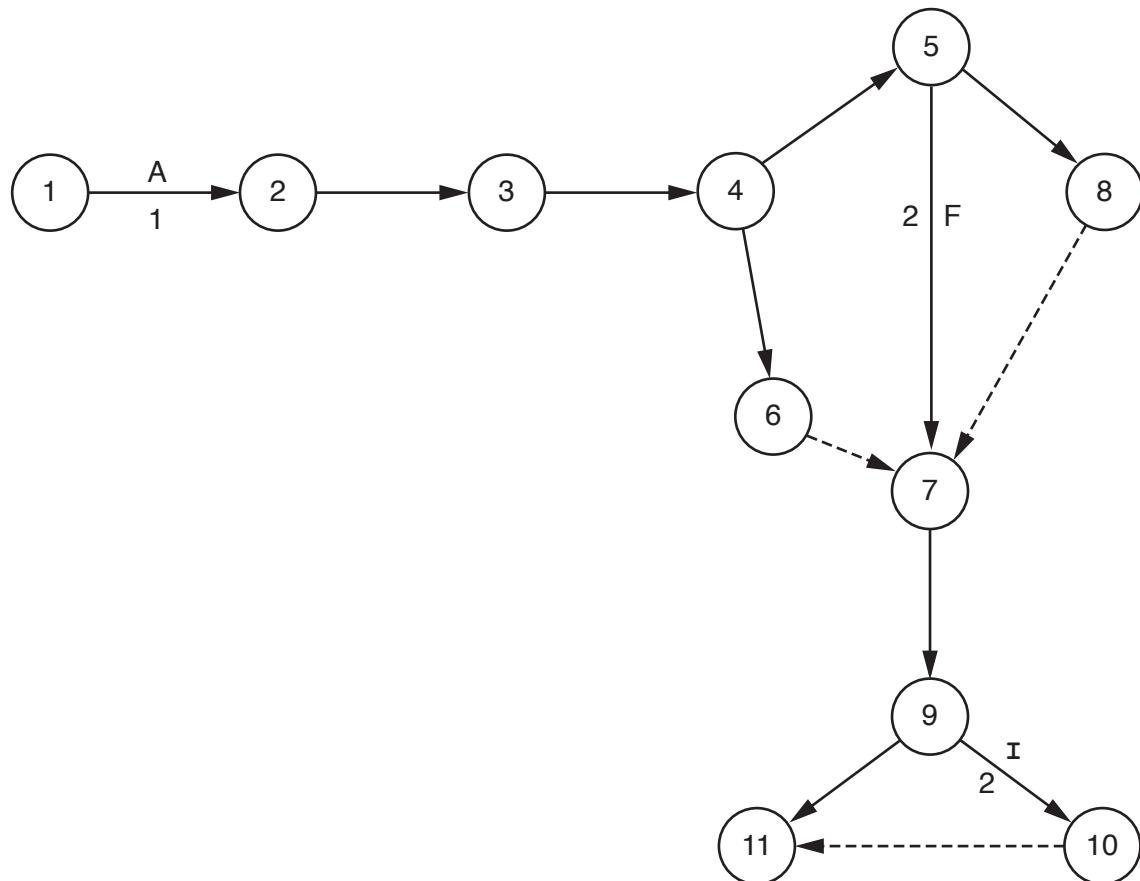
- 1 A technology company needs software to calculate how much each employee should be paid.

- (a) Developing the software will involve the following activities:

Activity	Description	Time to complete (weeks)	Predecessor
A	Identify requirements	1	–
B	Observe current system	1	A
C	Create algorithm design	3	B
D	Write code	10	C
E	Test modules	7	C
F	White box testing	2	D
G	Black box testing	3	D
H	Install software	1	E, F, G
I	Acceptance testing	2	H
J	Create user documentation	2	H

- (i) Add the correct activities and times to the following Program Evaluation Review Technique (PERT) chart for the software development.

Three of the activities and times have been done for you.



[7]

- (ii) The dashed line connecting nodes 10 and 11 indicates a dummy activity.

State the purpose of a dummy activity.

..... [1]

- (b) A bonus payment may be added to an employee's salary. A pension payment may also be subtracted from an employee's salary.

The company needs to assess what additions and subtractions should be made to the salary of each employee. There are three conditions to check:

- If the employee has worked a public holiday, they receive a 3% bonus payment.
- If the employee has worked 160 or more hours in a month, they receive an additional 5% bonus payment.
- If the employee pays into a pension, the company subtracts 4% for the pension payment.

Complete the decision table to show the additions and subtractions.

		Rules							
Conditions	Public holiday	Y	Y	Y	Y	N	N	N	N
	Hours \geq 160	Y	Y	N	N	Y	Y	N	N
	Pension	Y	N	Y	N	Y	N	Y	N
Actions	3% bonus payment								
	5% bonus payment								
	4% pension payment								

[3]

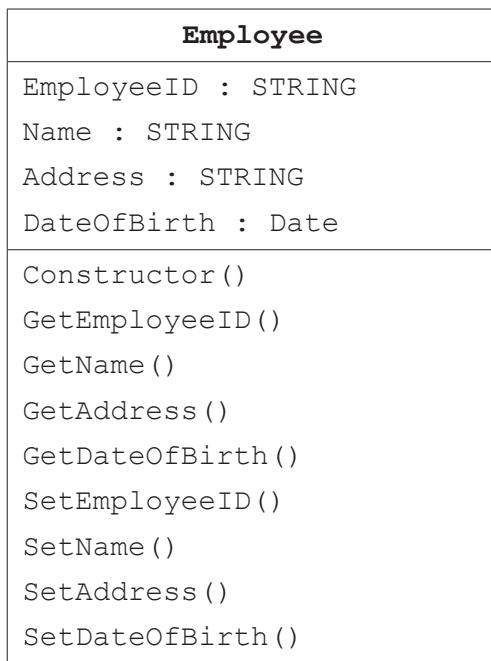
- (c) The company decides to implement a program for the software using object-oriented programming (OOP).

Each employee has a unique employee ID, name, address and date of birth. There are two types of employee: salary and apprenticeship.

Salaried employees are paid a fixed monthly payment. The hours a salary employee works in a month are recorded to calculate bonus payments. They may receive bonus payments and make pension payments (given in part(b)).

Apprenticeship employees are paid weekly. They receive an hourly rate of pay. Apprenticeship employees do not receive bonus payments or make pension payments.

- (i) Complete the following class diagram for the program.



[3]

- (ii) Write **program code** for the `Constructor()` in the `Employee` class.

All properties are sent as parameters.

Programming language

Program code

[4]

[4]

- (iii) Write **program code** for the `GetEmployeeID()` method in the `Employee` class.

The get method returns the value of the EmployeeID property.

Programming language

Program code

[2]

- (iv) Write program code for the SetEmployeeID() method in the Employee class.

The set method takes the new value as its parameter.

Programming language

Program code

[2]

[2]

- (v) Write program code for the SetPension() method in the SalaryEmployee() class.

- The method takes a new value for `Pension` as a parameter.
 - If the parameter's value is valid (it is `TRUE` or `FALSE`), the method returns `TRUE` and sets the parameter's value.
 - Otherwise the method returns `FALSE` and does not set `Pension`.

Programming language

Program code

[4]

Question 1 continues on the next page.

(vi) A SalaryEmployee is paid a fixed monthly payment.

- If the employee has worked a public holiday, they receive a 3% bonus payment. This is calculated from their `MonthlyPayment`.
 - If the employee has worked 160 or more hours in a month, they receive an additional 5% bonus payment, calculated from their `MonthlyPayment`.
 - If the employee pays into a pension, 4% will be subtracted from their `MonthlyPayment`.

Monthly salary is the final payment the employee receives.

For example, Chris is a `SalaryEmployee`. His `fixed MonthlyPayment` is \$1000. He has worked a public holiday. He has worked 165 hours this month. He pays into a pension.

- The public holiday bonus is \$30 (3% of \$1000)
 - The hours worked bonus payment is \$50 (5% of \$1000)
 - The pension payment is \$40 (4% of \$1000)

Chris's monthly salary is calculated as $(\$1000 + \$30 + \$50) - \$40 = \$1040$

The function `CalculateMonthlySalary()` is used to calculate the monthly salary. It:

- takes a `SalaryEmployee` as a parameter
 - calculates the bonus payments and pension payment
 - outputs the pension payment and total bonus payment
 - calculates and returns the monthly salary.

Write program code for the function CalculateMonthlySalary().

Programming language

Program code

- (d) Noona describes an example of a feature of object-oriented programming (OOP). She says:

"One method exists in the parent class but is overwritten in the child class, to behave differently."

Identify the feature Noona has described.

..... [1]

- 2 The number of cars that cross a bridge is recorded each hour. This number is placed in a circular queue before being processed.

- (a) The queue is stored as an array, NumberQueue, with eight elements. The function AddToQueue adds a number to the queue. EndPointer and StartPointer are global variables.

Complete the following **pseudocode** algorithm for the function AddToQueue.

```

FUNCTION AddToQueue (Number : INTEGER) RETURNS BOOLEAN

DECLARE TempPointer : INTEGER
CONSTANT FirstIndex = 0
CONSTANT LastIndex = .....
TempPointer ← EndPointer + 1
IF ..... > LastIndex
    THEN
        TempPointer ← .....
    ENDIF
    IF TempPointer = StartPointer
        THEN
            RETURN .....
        ELSE
            EndPointer ← TempPointer
            NumberQueue [EndPointer] ← .....
            RETURN TRUE
        ENDIF
    ENDFUNCTION

```

[5]

- (b) Describe how a number is removed from the circular queue to be processed.

.....
.....
.....
.....
.....
.....
.....
..... [4]

- (c) A queue is one example of an Abstract Data Type (ADT).

Identify **three other** Abstract Data Types.

- 1
2
3 [3]

- 3 A company wants to test a program to check that it works. They can use different types of test data to do this.

- (a) Identify **three** different types of test data that the company can use.

1

2

3

[3]

- (b) The programmer will make use of debugging features, when building and testing a program.

- (i) Two debugging features are described in the table.

Write the correct name for **each** debugging feature.

Description	Name of debugging feature
A point where the program can be halted to see if the program works to this point.
One statement is executed and then the program waits for input from the programmer to move on to the next statement.

[2]

- (ii) Identify **and** describe **one other** debugging feature.

Debugging feature

Description

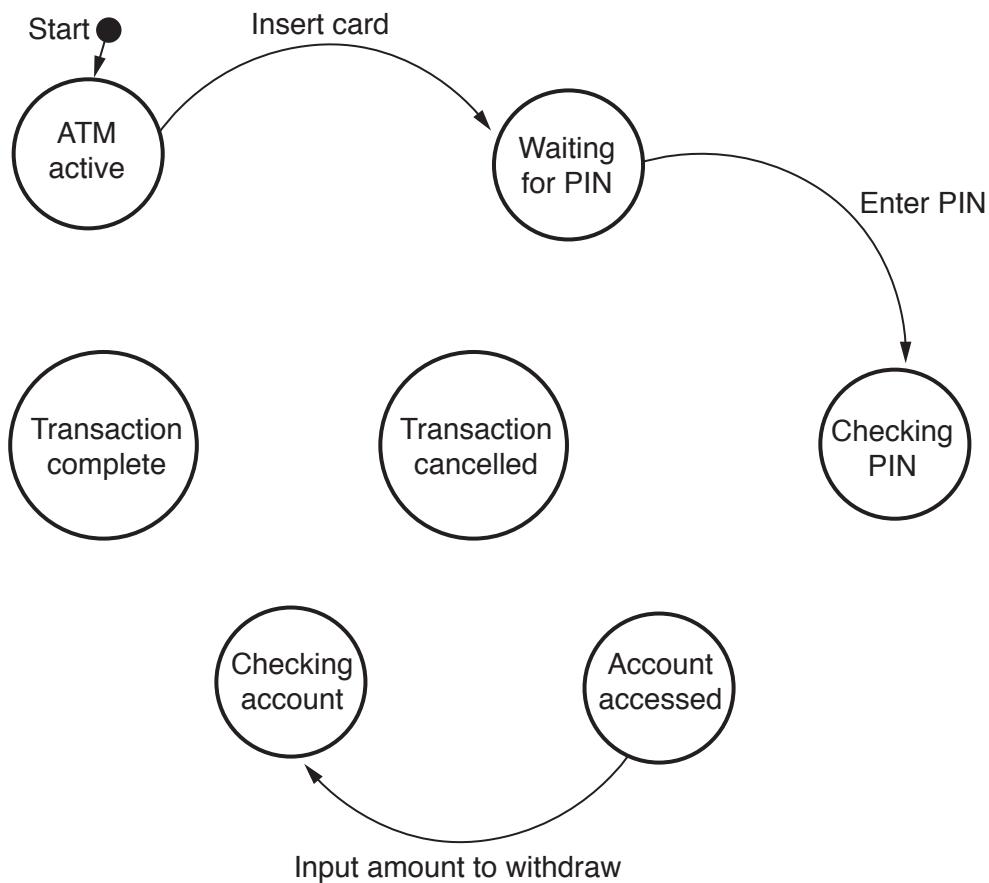
..... [2]

- 4 A bank wants to analyse how an automated teller machine (ATM) deals with transactions.

The following state-transition table shows the transitions from one state to another for a transaction.

Current state	Event	Next state
ATM active	Insert card	Waiting for PIN
Waiting for PIN	Enter PIN	Checking PIN
Waiting for PIN	Cancel selected	Transaction cancelled
Checking PIN	PIN valid	Account accessed
Checking PIN	PIN invalid	Waiting for PIN
Account accessed	Cancel selected	Transaction cancelled
Account accessed	Input amount to withdraw	Checking account
Checking account	Funds available	Transaction complete
Transaction complete	Return card and dispense cash	ATM active
Checking account	Funds not available	Account accessed
Transaction cancelled	Return card	ATM active

Complete the state-transition diagram to correspond with the table.



[8]

- 5 The following table shows part of the instruction set for a processor which has one general purpose register, the Accumulator (ACC) and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>. <address> can be an absolute address or a symbolic address.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

- (a) A programmer needs a program that multiplies a binary number by 4.

The programmer has started to write the program in the following table. The comment column contains explanations for the missing program instructions.

Write the program using the given instruction set.

Label	Instruction		Comment
	Op code	Operand	
			// load contents of NUMBER
			// perform shift to multiply by 4
			// store contents of ACC in NUMBER
			// end program
NUMBER:	B00110110		

[5]

Note:

- # denotes immediate addressing
- B denotes a binary number, e.g. B01001010
- & denotes a hexadecimal number, e.g. &4A

- (b) A programmer needs a program that counts the number of lower case letters in a string.

The programmer has started to write the program in the following table. The comment column contains explanations for the missing program instructions.

Complete the program using the given instruction set. A copy of the instruction set is provided on the opposite page.

Label	Instruction		Comment
	Op code	Operand	
	LDR	#0	// initialise Index Register to 0
START:			// load the next value from the STRING
			// perform bitwise AND operation with MASK
			// check if result is equal to MASK
			// if FALSE, jump to UPPER
			// increment COUNT
UPPER:	INC	IX	// increment the Index Register
			// decrement LENGTH
			// is LENGTH = 0 ?
			// if FALSE, jump to START
	END		// end program
MASK:	B00100000		// if bit 5 is 1, letter is lower case
COUNT:	0		
LENGTH:	5		
STRING:	B01001000		// ASCII code for 'H'
	B01100001		// ASCII code for 'a'
	B01110000		// ASCII code for 'p'
	B01110000		// ASCII code for 'p'
	B01011001		// ASCII code for 'Y'

[8]

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>. <address> can be an absolute address or a symbolic address.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/43

Paper 4 Practical

October/November 2023

2 hours 30 minutes

You will need: Candidate source files (listed on page 2)
evidence.doc



INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the evidence document, **evidence.doc**

Make sure that your name, centre number and candidate number appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: **evidence_zz999_9999**

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

One source file is used to answer **Question 2**. The file is called **QueueData.txt**

- 1 This iterative pseudocode algorithm for the function `IterativeVowels()` takes a string as a parameter and counts the number of lower-case vowels in this string.
The vowels are the letters a, e, i, o and u.

```
FUNCTION IterativeVowels(Value : STRING) RETURNS INTEGER
    DECLARE Total : INTEGER
    DECLARE LengthString : INTEGER
    DECLARE FirstCharacter : CHAR
    Total ← 0
    LengthString ← LENGTH(Value)
    FOR X ← 0 TO LengthString - 1
        FirstCharacter ← MID(Value, 0, 1)
        IF FirstCharacter = 'a' OR FirstCharacter = 'e' OR
            FirstCharacter = 'i' OR FirstCharacter = 'o' OR
            FirstCharacter = 'u' THEN
            Total ← Total + 1
        ENDIF
        Value ← MID(Value, 1, LENGTH(Value)-1)
    NEXT X
    RETURN Total
ENDFUNCTION
```

The pseudocode function `MID(X, Y, Z)` returns `Z` number of characters from string `X`, starting at the character in position `Y`. The first character in a string is in position 0, for example:

`MID("computer", 0, 3)` returns "com"

The pseudocode function `LENGTH(X)` returns the number of characters in the string `X`, for example:

`LENGTH("computer")` returns 8

- (a) (i) Write program code for the function `IterativeVowels()`.

Save your program as **Question1_N23**.

Copy and paste the program code into part **1(a)(i)** in the evidence document.

[5]

- (ii) Write program code to call the function `IterativeVowels()` with the parameter "house" from the main program.

Output the return value.

Save your program.

Copy and paste the program code into part **1(a)(ii)** in the evidence document.

[2]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **1(a)(iii)** in the evidence document.

[1]

- (b) (i) Rewrite the function `IterativeVowels()` as a recursive function with the identifier `RecursiveVowels()`.

Save your program.

Copy and paste the program code into part **1(b)(i)** in the evidence document.

[6]

- (ii) Write program code to call the function `RecursiveVowels()` with the parameter "imagine" from the main program.

Output the return value.

Save your program.

Copy and paste the program code into part **1(b)(ii)** in the evidence document.

[1]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **1(b)(iii)** in the evidence document.

[1]

2 A linear queue is implemented using the 1D array, Queue. The index of the first element in the array is 0.

(a) (i) Write program code to declare:

- Queue — a global array with space to store 50 IDs of type string
- HeadPointer — a global variable to point to the first element in the queue, initialised to -1
- TailPointer — a global variable to point to the next available space in the queue, initialised to 0.

Save your program as **Question2_N23**.

Copy and paste the program code into part **2(a)(i)** in the evidence document.

[2]

(ii) The procedure Enqueue () takes a string parameter.

If the queue is full, the procedure outputs a suitable message. If the queue is not full, the procedure inserts the parameter into the queue and updates the relevant pointer(s).

Write program code for Enqueue () .

Save your program.

Copy and paste the program code into part **2(a)(ii)** in the evidence document.

[4]

(iii) The function Dequeue () checks if the queue is empty.

If the queue is empty, the function outputs a suitable message and returns the string "Empty".

If the queue is not empty, the function returns the first element in the queue and updates the relevant pointer(s).

Write program code for Dequeue () .

Save your program.

Copy and paste the program code into part **2(a)(iii)** in the evidence document.

[4]

- (b) A shop sells computer games. Each game has a unique identifier (ID) of string data type.

The text file `QueueData.txt` contains a list of game IDs.

The procedure `ReadData()` reads the data from the text file and inserts each item of data into the array `Queue`.

Write program code for the procedure `ReadData()`.

Save your program.

Copy and paste the program code into part 2(b) in the evidence document.

[6]

- (c) Some game IDs appear in the text file more than once.

The program needs to total the number of times each game ID appears in the text file.

The record structure `RecordData` has the following fields:

- `ID` — a string to store the game ID
- `Total` — an integer to store the total number of times that game ID appears in the text file.

- (i) Write program code to declare the record structure `RecordData`.

If you are writing in Python, include attribute declarations as comments.

Save your program.

Copy and paste the program code into part 2(c)(i) in the evidence document.

[2]

- (ii) The global 1D array `Records` stores up to 50 items of type `RecordData`.

The global variable `NumberRecords` stores the number of records currently in the array `Records` and is initialised to 0.

Write program code to declare `Records` and `NumberRecords`.

If you are writing in Python, include attribute declarations as comments.

Save your program.

Copy and paste the program code into part 2(c)(ii) in the evidence document.

[2]

(iii) The pseudocode algorithm for the procedure TotalData ():

- uses Dequeue () to remove an ID from the queue
- checks whether a RecordData with the returned ID exists in Records
- increments the total for that ID in the record if the ID already exists
- creates a new record and stores it in Records if the ID does not exist.

```

PROCEDURE TotalData()

DECLARE DataAccessed : STRING
DECLARE Flag : BOOLEAN
DataAccessed ← Dequeue ()
Flag ← FALSE
IF NumberRecords = 0 THEN
    Records [NumberRecords].ID ← DataAccessed
    Records [NumberRecords].Total ← 1
    Flag ← TRUE
    NumberRecords ← NumberRecords + 1
ELSE
    FOR X ← 0 TO NumberRecords - 1
        IF Records [X].ID = DataAccessed THEN
            Records [X].Total ← Records [X].Total + 1
            Flag ← TRUE
        ENDIF
        NEXT X
    ENDIF
    IF Flag = FALSE THEN
        Records [NumberRecords].ID ← DataAccessed
        Records [NumberRecords].Total ← 1
        NumberRecords ← NumberRecords + 1
    ENDIF
ENDPROCEDURE

```

Write program code for the procedure TotalData ().

Save your program.

Copy and paste the program code into part 2(c)(iii) in the evidence document.

[5]

- (d) The procedure OutputRecords () outputs the ID and total of each record in Records in the format:

ID 1234 Total 4

Write program code for OutputRecords () .

Save your program.

Copy and paste the program code into part 2(d) in the evidence document.

[1]

- (e) The main program needs to:

- call ReadData ()
- call TotalData () for each element in the queue
- call OutputRecords () .

(i) Write program code for the main program.

Save your program.

Copy and paste the program code into part 2(e)(i) in the evidence document.

[2]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part 2(e)(ii) in the evidence document.

[1]

- 3 A computer game is written using object-oriented programming.

The game has multiple characters that can move around the screen.

The class `Character` stores data about the characters. Each character has a name, a current X (horizontal) position and a current Y (vertical) position.

Character	
Name : STRING	stores the name of the character as a string
XPosition : INTEGER	stores the X position as an integer
YPosition : INTEGER	stores the Y position as an integer
Constructor()	initialises Name, XPosition and YPosition to its parameter values
GetXPosition()	returns the X position
GetYPosition()	returns the Y position
SetXPosition()	adds the parameter to the X position validates that the new X position is between 0 and 10 000 inclusive
SetYPosition()	adds the parameter to the Y position validates that the new Y position is between 0 and 10 000 inclusive
Move()	takes a direction as a parameter and calls either SetXPosition or SetYPosition with an integer value

- (a) (i) Write program code to declare the class `Character` and its constructor.

Do **not** declare the other methods.

Use your programming language's appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_N23**.

Copy and paste the program code into part 3(a)(i) in the evidence document.

[4]

- (ii) The get methods `GetXPosition()` and `GetYPosition()` each return the relevant attribute.

Write program code for the get methods.

Save your program.

Copy and paste the program code into part 3(a)(ii) in the evidence document.

[3]

- (iii) The set methods `SetXPosition()` and `SetYPosition()` each take a value as a parameter and add this to the current X or Y position.

If the new value exceeds 10 000, it is limited to 10 000.

If the new value is below 0, it is limited to 0.

Write program code for the set methods.

Save your program.

Copy and paste the program code into part 3(a)(iii) in the evidence document.

[4]

- (iv) The method `Move()` takes a string parameter: "up", "down", "left" or "right".

The table shows the change each direction will make to the X or Y position.

Use the appropriate method to change the position value.

Direction	Value change
up	Y position + 10
down	Y position - 10
left	X position - 10
right	X position + 10

Write program code for `Move()`.

Save your program.

Copy and paste the program code into part 3(a)(iv) in the evidence document.

[4]

- (b) Write program code to declare a new instance of `Character` with the identifier `Jack`.

The starting X position is 50 and the starting Y position is 50, the character's name is Jack.

Save your program.

Copy and paste the program code into part 3(b) in the evidence document.

[2]

- (c) The class `BikeCharacter` inherits from the class `Character`.

BikeCharacter	
<code>Constructor()</code>	takes <code>Name</code> , <code>XPosition</code> and <code>YPosition</code> as parameters calls its parent class constructor with the appropriate values
<code>Move()</code>	overrides the method <code>Move()</code> from the parent class by changing either the X position or the Y position by 20 instead of 10

- (i) Write program code to declare the class `BikeCharacter` and its constructor.

Do **not** declare the other method.

Use your programming language's appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into part 3(c)(i) in the evidence document.

[3]

- (ii) The method `Move()` overrides the method from the parent class.

The table shows the change each direction will make to the X or Y position.

Direction	Value change
up	Y position + 20
down	Y position - 20
left	X position - 20
right	X position + 20

Write program code for `Move()`.

Save your program.

Copy and paste the program code into part 3(c)(ii) in the evidence document.

[2]

- (d) Write program code to declare a new instance of `BikeCharacter` with the identifier `Karla`.

The starting X position is 100, the starting Y position is 50 and the character's name is Karla.

Save your program.

Copy and paste the program code into part 3(d) in the evidence document.

[1]

(e) (i) Write program code to:

- take as input which of the two characters the user would like to move
- take as input the direction the user would like the character to move
- call the appropriate method to move the character
- output the character's new X and Y position in an appropriate format, for example:
"Karla's new position is X = 100 Y = 200"

All inputs require appropriate prompts and must be validated.

Save your program.

Copy and paste the program code into part 3(e)(i) in the evidence document.

[5]

(ii) Test your program twice with the following inputs.

Test 1: jack right

Test 2: karla down

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part 3(e)(ii) in the evidence document.

[2]

BLANK PAGE

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.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

May/June 2017

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **15** printed pages and **1** blank page.

- 1 The following table shows part of the instruction set for a processor which has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
STO	<address>	Store the contents of ACC at the given address.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
CMP	<address>	Compare the contents of ACC with the contents of <address>.
JMP	<address>	Jump to given address.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

- (a) A programmer writes a program that:

- reads a character from the keyboard (assume it will be a capital letter)
- outputs the alphabetical sequence of characters from 'A' to the character input. For example, if the character 'G' is input, the output is:

ABCDEFG

The programmer has started to write the program in the table on the following page. The Comment column contains descriptions for the missing instructions, labels and data.

Complete the following program. Use op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// INPUT character
			// store in CHAR
			// Initialise ACC (ASCII value for 'A' is 65)
			// OUTPUT ACC
			// compare ACC with CHAR
			// if equal jump to end of FOR loop
			// increment ACC
			// jump to LOOP
ENDFOR:	END		
CHAR:			

[8]

(b) The programmer now starts to write a program that:

- tests whether an 8-bit two's complement integer stored at address NUMBER is positive or negative
- outputs 'P' for a positive integer and 'N' for a negative integer.

Complete the following program. Use op codes from the given instruction set.
Show the required value of MASK in binary.

Label	Op code	Operand	Comment
START:			
		MASK	// set to zero all bits except sign bit
			// compare with 0
			// if not equal jump to ELSE
THEN:			// load ACC with 'P' (ASCII value 80)
	JMP	ENDIF	
ELSE:			// load ACC with 'N' (ASCII value 78)
ENDIF:			
	END		
NUMBER:	B00000101		// integer to be tested
MASK:			// value of mask in binary

[7]

- 2** A hash table has these associated operations:

- create hash table
- insert record
- search hash table

A hash table is to be used to store customer records.

Each record consists of a unique customer ID, the record key, and other customer data.

- (a)** The following pseudocode declares a customer record structure.

```
TYPE CustomerRecord
    CustomerID : INTEGER
    Data : STRING
ENDTYPE
```

The hash table is to be implemented as a 1D array `Customer` with elements indexed 0 to 199. The procedure to create a hash table will declare and initialise the array by storing 200 records with the `CustomerID` field in each record set to 0.

Complete the **pseudocode**.

```
PROCEDURE CreateHashTable()
```

```
ENDPROCEDURE
```

[4]

- (b)** A hashing function `Hash` exists, which takes as a parameter the customer ID and returns an integer in the range 0 to 199 inclusive.

- (i)** The procedure, `InsertRecord`, takes as a parameter the customer record to be inserted into the hash table.

The procedure makes use of the function `Hash`. Collisions will be managed using open hashing. This means a collision is resolved by storing the record in the next available location. The procedure will generate an error message if the hash table is full.

Complete the **pseudocode** for the procedure.

```

PROCEDURE InsertRecord(BYVALUE NewCustomer : CustomerRecord)

    TableFull ← FALSE

    // generate hash value

    Index ← .....

    Pointer ← Index    // initialise Pointer variable to hash value

    // find a free table element

    WHILE .....

        Pointer ← .....

        // wrap back to beginning of table if necessary

        IF .....

            THEN
                .....
            ENDIF

        // check if back to original index

        IF .....

            THEN
                TableFull ← TRUE
            ENDIF

        ENDWHILE

        IF .....

            THEN
                .....
            ELSE
                .....
            ENDIF
        ENDIF
    ENDPROCEDURE

```

[9]

- (ii) The function `SearchHashTable` will search for a record in the hash table. The function takes as a parameter the customer ID to be searched for. The function will return the position in the hash table where the record has been saved. If the hash table does not contain the record, the function will return the value -1.

You can assume that there is at least one empty record in the hash table.

Complete the **pseudocode** for the function.

```

FUNCTION SearchHashTable (BYVALUE SearchID : INTEGER) RETURNS INTEGER

    // generate hash value
    Index ← .....  

    .....  

    // check each record from index until found or not there
    WHILE ( ..... )  

        AND (.....)  

        .....  

        .....  

    // wrap if necessary
    IF .....  

        THEN  

        .....  

    ENDIF  

    ENDWHILE  

    // has customer ID been found?  

    IF .....  

        THEN  

        .....  

    ELSE  

        .....  

    ENDIF  

ENDFUNCTION

```

[9]

- (iii) A record that is no longer required is deleted.

State the problem that might be caused by this deletion.

.....
.....

[1]

- 3 NameList is a 1D array that stores a sorted list of names. A programmer declares the array in pseudocode as follows:

```
NameList : Array[0 : 100] OF STRING
```

The programmer wants to search the list using a binary search algorithm.

The programmer decides to write the search algorithm as a recursive function. The function, Find, takes three parameters:

- Name, the string to be searched for
- Start, the index of the first item in the list to be searched
- Finish, the index of the last item in the list to be searched

The function will return the position of the name in the list, or -1 if the name is not found.

Complete the **pseudocode** for the recursive function.

```
FUNCTION Find(BYVALUE Name : STRING, BYVALUE Start : INTEGER,
              BYVALUE Finish : INTEGER) RETURNS INTEGER

    // base case

    IF ..... THEN
        RETURN -1
    ELSE
        Middle ← .....
        IF ..... THEN
            RETURN .....
        ELSE // general case
            IF SearchItem > .....
                THEN
                    .....
                ELSE
                    .....
            ENDIF
        ENDIF
    ENDIF
ENDFUNCTION
```

4 An ordered linked list Abstract Data Type (ADT) has these associated operations:

- create list
- add item to list
- output list to console

The ADT is to be implemented using object-oriented programming as a linked list of nodes.

Each node consists of data and a pointer.

(a) There are two classes, `LinkedList` and `Node`.

(i) State the term used to describe the relationship between these classes.

..... [1]

(ii) Draw the appropriate diagram to represent this relationship. Do not list the attributes and methods of the classes.

[2]

(b) The design for the Node class consists of:

- attributes
 - Data : STRING
 - Pointer : INTEGER
 - methods
 - CreateNode(Data, Pointer)
 - SetData(Data)
 - SetPointer(Pointer)
 - GetData() RETURNS STRING
 - GetPointer() RETURNS INTEGER

The constructor method sets the attributes to the initial values that are passed as parameters.

Write program code for:

- the `Node` class declaration
 - the constructor.

Programming language used

Program code

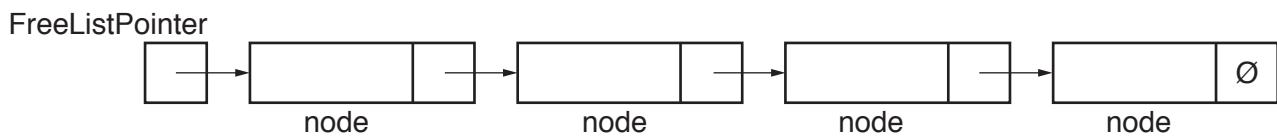
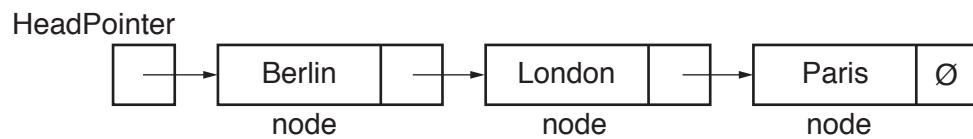
[5]

. [5]

(c) The identifier table for the `LinkedList` class is:

Identifier	Data type	Description
HeadPointer	INTEGER	Pointer to the first node in the ordered list.
FreeListPointer	INTEGER	Pointer to the first node in the free list.
NodeArray	ARRAY [0 : 7] OF Node	1D array stores the nodes that make the ordered linked list. The unused nodes are linked together into a free list.
Constructor()		Constructor instantiates an object of <code>LinkedList</code> class, initialises <code>HeadPointer</code> to be a null pointer and links all nodes to form the free list.
FindInsertionPoint()		Procedure that takes the new data item as the parameter <code>NewData</code> and returns two parameters: <ul style="list-style-type: none"> • <code>PreviousPointer</code>, whose value is: <ul style="list-style-type: none"> ◦ either pointer to node before the insertion point ◦ or the null pointer if the new node is to be inserted at the beginning of the list. • <code>NextPointer</code>, whose value is a pointer to node after the insertion point.
AddToList (NewString)		Procedure that takes as a parameter a unique string and links it into the correct position in the ordered list.
OutputListToConsole()		Procedure to output all the data from the list pointed to by <code>HeadPointer</code> .

The following diagram shows an example of a linked list object. This example list consists of three nodes, linked in alphabetical order of the data strings. The unused nodes are linked to form a free list.



The symbol \emptyset represents a null pointer.

(i) Explain the meaning of the term **null pointer**.

- (ii) Give an appropriate value to represent the null pointer for this design. Justify your answer.

[2]

- [2]

- (iii) Write **program code** for the `LinkedList` class declaration **and** the constructor.

Programming language used

Program code

[7]

- (iv) Write **program code** to instantiate a linked list object with the `contacts` identifier.

Programming language used

Program code

[1]

- (v) The `OutputListToConsole` method is to output all the data stored in the linked list. `HeadPointer` points to the first list node.

Write **program code** for this method.

Programming language used

Program code

. [5]

Question 4 continues on page 14.

(vi) The structured English for the AddToList (NewString) method is as follows:

```

    Make a copy of the value of free list pointer, name it NewNodePointer

    Store new data item in free node pointed to by NewNodePointer

    Adjust free list pointer to point to next free node

    IF linked list is currently empty

        THEN

            Make this node the first node

            Set pointer of this node to null pointer

        ELSE

            Find insertion point using the FindInsertionPoint method

            // FindInsertionPoint provides

            // pointer to previous node and pointer to next node

            IF previous pointer is null pointer

                THEN

                    Link this node to front of list

                ELSE

                    Link this node between previous node and next node
    
```

The FindInsertionPoint method receives the new data item as the parameter NewString. It returns two parameters:

- PreviousPointer, whose value is:
 - either the pointer to the node before the insertion point
 - or the null pointer, if the new node is to be inserted at the beginning of the list.
- NextPointer, whose value is the pointer to the node after the insertion point.

Write program code for the AddToList method.

Programming language used

Program code

[6]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/42

Paper 4 Practical

May/June 2023

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages.

Open the document **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

Two source files are used to answer **Question 3**. The files are called **Employees.txt** and **HoursWeek1.txt**

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

1 A 1D array needs to store the names of 10 animals.

(a) Write program code to declare the global string array **Animals** to store 10 items.

Save your program as **Question1_J2023**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[2]

(b) The main program needs to store the following animals in the array:

horse
lion
rabbit
mouse
bird
deer
whale
elephant
kangaroo
tiger

Write program code to store these animal names in the array.

The names must be in lower case and stored in the order given in the list.

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[2]

- (c) The following pseudocode procedure sorts the array into a descending alphabetical order using only the first character in each animal name.

The function LENGTH (DataArray) returns the number of elements in the array DataArray.

The function MID(String, Start, Quantity) returns Quantity number of characters from String starting at index Start. The first character in the string is index 0, for example:

MID("tiger", 0, 2) will return "ti"

There are **four** incomplete statements in the procedure.

```
PROCEDURE SortDescending()

DECLARE ArrayLength : INTEGER

DECLARE Temp : STRING

ArrayLength ← LENGTH(Animals)

FOR X ← 0 TO ArrayLength - 1

    FOR Y ← ..... TO ArrayLength - X - 1

        IF MID(Animals[Y], 0, 1) < MID(Animals[.....], 0, 1) THEN

            Temp ← Animals[.....]

            Animals[Y] ← Animals[Y + 1]

            Animals[Y + 1] ← ......

        ENDIF

    NEXT Y

NEXT X

ENDPROCEDURE
```

Write program code for the procedure SortDescending().

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[6]

(d) (i) Write program code to amend the main program to:

- call the procedure `SortDescending()`
- output the sorted contents of the array with each animal name on a new line.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[3]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 1(d)(ii)** in the evidence document.

[1]

2 A business sells a single product. Customers can purchase one or more of this product.

Each sale has an ID and a quantity, for example "ABC" and 2

The business needs a program to store the data about the sales in a circular queue data structure.

- (a) Write program code to declare a record structure, SaleData, to store the data about each sale.

Save your program as **Question2_J2023**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[2]

- (b) Write program code to:

- declare a global array, CircularQueue, of 5 items to store the sale records
- declare the global pointers Head and Tail
- declare the global variable NumberOfItems
- initialise all elements of the array CircularQueue to an empty record, where the ID is null ("") and quantity is -1
- initialise Head, Tail and NumberOfItems to 0

Save your program.

Copy and paste the program code into **part 2(b)** in the evidence document.

[4]

- (c) The function Enqueue () :

- takes a new record as a parameter
- inserts the record in the circular queue at the element pointed to by Tail
- updates pointers and other variables as required
- returns -1 if the circular queue is full
- returns 1 if the record is stored successfully.

Write program code for the function Enqueue () .

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[6]

(d) The function `Dequeue ()`:

- returns a null or empty record if the circular queue is empty
- returns the first record in the queue if the circular queue is not empty
- updates pointers and other variables as required.

Write program code for the function `Dequeue ()`.

Save your program.

Copy and paste the program code into **part 2(d)** in the evidence document.

[6]

(e) The procedure `EnterRecord ()`:

- takes an ID and quantity as input and creates a sale record
- uses `Enqueue ()` to insert the record in the circular queue
- outputs "Full" if the record was not inserted in the circular queue
- outputs "Stored" if the record was inserted in the circular queue.

Write program code for the procedure `EnterRecord ()`.

Save your program.

Copy and paste the program code into **part 2(e)** in the evidence document.

[5]

(f) The following sale records need to be entered into the program:

ID	Quantity
ADF	10
OOP	1
BXW	5
XXZ	22
HQR	6
LLP	3

(i) Amend the main program to:

- use `EnterRecord()` to input the six records in the table
- use `Dequeue()` to remove one record
- output either the ID and quantity of the removed record, or an error message if the circular queue is empty
- use `EnterRecord()` to input the record with the ID "LLP" for a second time
- output the ID and quantity for all the records currently stored in the array `CircularQueue`.

Write program code to perform these tasks.

Save your program.

Copy and paste the program code into **part 2(f)(i)** in the evidence document.

[4]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 2(f)(ii)** in the evidence document.

[1]

- 3 A company needs a computer program to store data about its employees.

Part of the program is being written using object-oriented programming.

The class `Employee` stores data about the employees. Each employee has an employee number, a job title and hourly pay rate. The class will also store the amount they are paid each week over a 52-week year in a 1D array.

Employee	
<code>HourlyPay : REAL</code>	stores the amount each employee gets paid each hour
<code>EmployeeNumber : STRING</code>	stores the employee's unique number
<code>JobTitle : STRING</code>	stores the employee's job title
<code>PayYear2022 : ARRAY[0:51] OF REAL</code>	stores the amount the employee has been paid each week
<code>Constructor()</code>	initialises <code>HourlyPay</code> , <code>EmployeeNumber</code> and <code>JobTitle</code> from the values passed as parameters initialises all 52 elements in <code>PayYear2022</code> to 0.0
<code>GetEmployeeNumber()</code>	returns the employee number
<code>SetPay()</code>	takes the week number and number of hours worked that week as parameters calculates and stores the pay for that week in <code>PayYear2022</code>
<code>GetTotalPay()</code>	returns the total of all the values in <code>PayYear2022</code>

- (a) (i) Write program code to declare the class `Employee`.

You only need to declare the class and its constructor. Do **not** declare any other methods.

Use your programming language appropriate constructor.

If you are writing program code in Python, include attribute declarations using comments.

Save your program as **Question3_J2023**.

Copy and paste the program code into **part 3(a)(i)** in the evidence document.

[5]

- (ii) The method `GetEmployeeNumber()` returns the employee number.

Write program code for the method `GetEmployeeNumber()`.

Save your program.

Copy and paste the program code into **part 3(a)(ii)** in the evidence document.

[2]

(iii) The method `SetPay()`:

- takes a week number and the number of hours worked that week as parameters
- calculates the pay for that week by multiplying the hourly pay by the number of hours worked that week
- stores the calculated pay in the appropriate index for that week in `PayYear2022`.

Write program code for the method `SetPay()`.

Save your program.

Copy and paste the program code into **part 3(a)(iii)** in the evidence document.

[3]

(iv) The method `GetTotalPay()` returns the total of all the values in `PayYear2022`.

Write program code for the method `GetTotalPay()`.

Save your program.

Copy and paste the program code into **part 3(a)(iv)** in the evidence document.

[2]

- (b) The child class Manager inherits from the parent class Employee.

A manager gets a bonus. This bonus value is a percentage, for example 10.0%.

When calculating the pay, the number of hours the manager worked that week is increased by the bonus value.

Manager	
BonusValue : REAL	stores the bonus value, for example 10.0 represents a 10.0% increase
Constructor()	takes bonus value, hourly pay, employee number and job title as parameters initialises BonusValue to its parameter value
SetPay()	takes the week number and number of hours worked as parameters increases the number of hours worked by the bonus value calls the SetPay() method from the parent class

- (i) Write program code to declare the class Manager.

You only need to declare the class and its constructor. Do **not** declare any other methods.

Use your programming language appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into **part 3(b)(i)** in the evidence document.

[4]

- (ii) The Manager method SetPay() overrides the method from the parent class and:

- takes the week number and number of hours worked as parameters
- increases the number of hours worked by the bonus value
- calls SetPay() from the parent class.

Write program code for the method SetPay().

Save your program.

Copy and paste the program code into **part 3(b)(ii)** in the evidence document.

[3]

- (c) The main program has a global 1D array, `EmployeeArray`, to store data about eight employees. Each employee is stored as an object of type `Employee`.

The file `Employees.txt` stores data about the employees, in the order:

- hourly pay
- employee number
- bonus value (where included)
- job title.

Only employees who are managers have a bonus value saved. For example:

- The first employee is a Junior Developer, with employee number 12452 and an hourly pay of \$15.22. This employee does not have a bonus value.
- The third employee is an Interface Manager, with employee number 02586 and an hourly pay of \$22.50. This employee has a bonus value of 5.25%.

Write the main program to:

- declare the array to store data about 8 employees
- read in the data from the file for each employee
- instantiate each employee as either `Employee` (if the employee does not have a bonus value) or `Manager` (if the employee has a bonus value).

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[7]

- (d) The file `HoursWeek1.txt` stores the number of hours each employee has worked in week 1, in the order:

- employee number
- number of hours worked.

For example, the first set of data is for employee 21548 who has worked 50.0 hours.

The procedure `EnterHours()`:

- reads in the values from the file
- finds the location of each employee in `EmployeeArray`
- calls the method `SetPay()` for each employee.

Write program code for `EnterHours()`.

Save your program.

Copy and paste the program code into **part 3(d)** in the evidence document.

[4]

- (e) (i) The main program needs to call `EnterHours()` and use the method `GetTotalPay()` to output the employee number and total pay for each of the eight employees.

Amend the main program to perform these tasks.

Save your program.

Copy and paste the program code into **part 3(e)(i)** in the evidence document.

[2]

- (ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 3(e)(ii)** in the evidence document.

[1]

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.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/42

Paper 4 Practical

October/November 2022

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the evidence document, **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: **evidence_zz999_9999**

A class declaration can be used to declare a record.

If the programming language does not support arrays, a list can be used instead.

A source file is used to answer **Question 2**. The file is called **Characters.txt**

- 1 A computer program is needed to store jobs in order of priority. Each job has a job number (for example, 123) and a priority from 1 to 10, with 1 being the highest priority and 10 the lowest.

The program stores the jobs in a global 2D array.

The pseudocode declaration for the array is:

```
DECLARE Jobs : ARRAY[0:99, 0:1] OF INTEGER
```

For example:

- `Jobs [0, 0]` stores the job number of the first job.
- `Jobs [0, 1]` stores the priority of the first job.

The global variable, `NumberOfJobs`, stores the number of jobs currently in the array.

- (a) Write program code to declare the global 2D array `Jobs` and the global variable `NumberOfJobs`.

Save your program as **Question1_N22**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[3]

- (b) The procedure `Initialise()` stores `-1` in each of the array elements and assigns `0` to `NumberOfJobs`.

Write program code for the procedure `Initialise()`.

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[3]

(c) The procedure `AddJob()`:

- takes a job number and priority as parameters
- stores the job in the next free array element
- outputs 'Added' if the job was successfully stored in the array
- outputs 'Not added' if the job was not successfully stored in the array.

Write program code for the procedure `AddJob()`.

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[5]

(d) The main program should call the procedure `Initialise()` and then use the `AddJob()` procedure to store the following jobs in the order given:

Job number	Priority
12	10
526	9
33	8
12	9
78	1

Write program code for the main program and perform the tasks described.

Save your program.

Copy and paste the program code into **part 1(d)** in the evidence document.

[2]

(e) When a new job has been added, the array is sorted into ascending numerical order of priority using an insertion sort.

Write program code for the procedure `InsertionSort()` to sort the data into ascending numerical order of priority.

Save your program.

Copy and paste the program code into **part 1(e)** in the evidence document.

[5]

- (f) The procedure `PrintArray()` outputs each job number and priority on a line, for example:

123 priority 1

39 priority 3

120 priority 7

Write program code for the procedure `PrintArray()`.

Save your program.

Copy and paste the program code into **part 1(f)** in the evidence document.

[3]

- (g) The main program needs to sort the array and then output the contents of the array.

- (i) Amend the main program by writing program code to call procedures `InsertionSort()` and `PrintArray()`.

Save your program.

Copy and paste the program code into **part 1(g)(i)** in the evidence document.

[1]

- (ii) Test your program.

Take a screenshot of the output.

Copy and paste the screenshot into **part 1(g)(ii)** in the evidence document.

[1]

- 2 A computer game is being developed. The game has 10 different characters that are all active in the game.

Part of the game is being written using object-oriented programming.

The class Character stores data about the characters. Each character has a name and the x coordinate and y coordinate of their current position.

Character	
Name : STRING	stores the name of the character
XCoordinate : INTEGER	stores the x coordinate
YCoordinate : INTEGER	stores the y coordinate
Constructor()	initialises Name, XCoordinate and YCoordinate from the values passed as parameters
GetName()	returns the name of the character
GetX()	returns the x coordinate of the character
GetY()	returns the y coordinate of the character
ChangePosition()	takes XChange as an integer parameter and adds it to the x coordinate takes YChange as an integer parameter and adds it to the y coordinate

- (a) Write program code to declare the class Character and its constructor. Do **not** write program code for the other methods.

Use your programming language appropriate constructor.

All attributes must be private. If you are writing in Python, include attribute declarations using comments.

Save your program as **Question2_N22**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[4]

- (b) Write program code for the **three** get methods for the class Character.

Save your program.

Copy and paste the program code into **part 2(b)** in the evidence document.

[3]

(c) Write program code for the method `ChangePosition()`.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[2]

(d) The main program has a 1D array of characters. Each character is stored as an object of type `Character`.

The game has a maximum of 10 characters. The character names, x coordinates and y coordinates are stored in the file `Characters.txt` in the order:

- name
- x coordinate
- y coordinate.

For example, the first character in the file is named Amal, with the x coordinate 0 and the y coordinate 2.

Amend the main program by writing program code to:

- declare the array
- read in all 10 characters from `Characters.txt`
- store each character as an object in the array.

Save your program.

Copy and paste the program code into **part 2(d)** in the evidence document.

[7]

(e) The main program needs to read in a character's name from the user, search for the character in the array and store the index of its position. It repeats until the user enters a name that exists in the array.

Amend the main program by writing program code to perform this task.

Save your program.

Copy and paste the program code into **part 2(e)** in the evidence document.

[5]

- (f) The user will enter a letter to identify the direction the chosen character from **part 2(e)** should move.

- If 'A' is input, the character moves left (x coordinate minus 1).
- If 'W' is input, the character moves up (y coordinate plus 1).
- If 'S' is input, the character moves down (y coordinate minus 1).
- If 'D' is input, the character moves right (x coordinate plus 1).

Amend the main program by writing program code to:

- take a letter as input until it is a valid move (A, W, S or D)
- change the position of the character using the appropriate method.

Save your program.

Copy and paste the program code into **part 2(f)** in the evidence document.

[7]

- (g) (i) When a change to a character's position has been made, the program needs to output the character's name and the new x and y coordinates of the character, in the format:

Qui has changed coordinates to X = 83 and Y = 0

Amend the main program by writing program code to perform these tasks.

Save your program.

Copy and paste the program code into **part 2(g)(i)** in the evidence document.

[2]

- (ii) Test your program by inputting the following **four** items of data in the order given:

THOMAS
qui
X
A

Take a screenshot of the output.

Copy and paste the screenshot into **part 2(g)(ii)** in the evidence document.

[1]

3 A program uses a linear queue to store up to 100 integers.

- (a) A 1D array, `Queue`, is used to store the data. The head pointer points to the first number stored in the queue and the tail pointer points to the next free space in the queue.

Write program code to:

- declare the global array `Queue`
- declare the global variable head pointer and assign an appropriate initial value
- declare the global variable tail pointer and assign an appropriate initial value.

Save your program as **Question3_N22**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[3]

- (b) The function `Enqueue()` takes an integer value as a parameter and stores it in the queue. It returns `TRUE` if the value was successfully stored and `FALSE` otherwise.

Write program code for the function `Enqueue()`.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[6]

- (c) The main program uses the `Enqueue()` function to store the numbers 1 to 20 (inclusive) in the queue, in ascending numerical order. The program should output 'Successful' if all numbers are successfully enqueued, and 'Unsuccessful' otherwise.

Amend the main program by writing program code to perform this task.

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[4]

- (d) The following iterative pseudocode function calculates the total of all the values stored in the queue.

```

FUNCTION IterativeOutput(Start: INTEGER) RETURNS INTEGER

DECLARE Total : INTEGER

Total ← 0

FOR Count ← Start - 1 TO HeadPointer STEP -1

    Total ← Total + Queue[Count]

NEXT Count

RETURN Total

ENDFUNCTION

```

Rewrite the function as a recursive function using program code.

Save your program.

Copy and paste the program code into **part 3(d)** in the evidence document.

[6]

- (e) The main program calls the recursive function from **part 3(d)** and outputs the value returned.

- (i) Amend the main program by writing program code to perform this task.

Save your program.

Copy and paste the program code into **part 3(e)(i)** in the evidence document.

[1]

- (ii) Test your program.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 3(e)(ii)** in the evidence document.

[1]

BLANK PAGE

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.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/43

Paper 4 Practical

May/June 2021

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the document **evidence.doc**.

Make sure that your name, centre number and candidate number will appear on every page of this document. This document will contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

A list is an alternative to an array.

A source file is used to answer question 3. The file is called **TreasureChestData.txt**

- 1 An unordered linked list uses a 1D array to store the data.

Each item in the linked list is of a record type, node, with a field data and a field nextNode.

The current contents of the linked list are:

startPointer	0	Index	data	nextNode
emptyList	5	0	1	1
		1	5	4
		2	6	7
		3	7	-1
		4	2	2
		5	0	6
		6	0	8
		7	56	3
		8	0	9
		9	0	-1

- (a) The following is pseudocode for the record type node.

```

TYPE node
  DECLARE data : INTEGER
  DECLARE nextNode : INTEGER
ENDTYPE

```

Write program code to declare the record type node.

Save your program as **question1**.

Copy and paste the program code into **part 1(a)** in the evidence document.

- (b) Write program code for the main program.

Declare a 1D array of type `node` with the identifier `linkedList`, and initialise it with the data shown in the table on page 2. Declare the pointers.

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[4]

- (c) The procedure `outputNodes()` takes the array and `startPointer` as parameters. The procedure outputs the data from the linked list by following the `nextNode` values.

- (i) Write program code for the procedure `outputNodes()`.

Save your program.

Copy and paste the program code into **part 1(c)(i)** in the evidence document.

[6]

- (ii) Edit the main program to call the procedure `outputNodes()`.

Take a screenshot to show the output of the procedure `outputNodes()`.

Save your program.

Copy and paste the screenshot into **part 1(c)(ii)** in the evidence document.

[1]

- (d) The function, `addNode()`, takes the linked list and pointers as parameters, then takes as input the data to be added to the end of the `linkedList`.

The function adds the node in the next available space, updates the pointers and returns True. If there are no empty nodes, it returns False.

- (i) Write program code for the function `addNode()`.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[7]

- (ii) Edit the main program to:

- call `addNode()`
- output an appropriate message depending on the result returned from `addNode()`
- call `outputNodes()` twice; once before calling `addNode()` and once after calling `addNode()`.

Save your program.

Copy and paste the program code into **part 1(d)(ii)** in the evidence document.

[3]

- (iii) Test your program by inputting the data value 5 and take a screenshot to show the output.

Save your program.

Copy and paste the screenshot into **part 1(d)(iii)** in the evidence document.

[1]

BLANK PAGE

2 A program stores the following ten integers in a 1D array with the identifier `arrayData`.

10	5	6	7	1	12	13	15	21	8
----	---	---	---	---	----	----	----	----	---

(a) Write program code for a **new program** to:

- declare the global 1D array, `arrayData`, with ten elements
- initialise `arrayData` in the main program using the data values shown.

Save your program as **question2**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[2]

(b) (i) A function, `linearSearch()`, takes an integer as a parameter and performs a linear search on `arrayData` to find the parameter value. It returns `True` if it was found and `False` if it was not found.

Write program code for the function `linearSearch()`.

Save your program.

Copy and paste the program code into **part 2(b)(i)** in the evidence document.

[6]

(ii) Edit the main program to:

- allow the user to input an integer value
- pass the value to `linearSearch()` as the parameter
- output an appropriate message to tell the user whether the search value was found or not.

Save your program.

Copy and paste the program code into **part 2(b)(ii)** in the evidence document.

[4]

(iii) Test your program with one value that is in the array and one value that is not in the array.

Take a screenshot to show the result of each test.

Save your program.

Copy and paste the screenshots into **part 2(b)(iii)** in the evidence document.

[2]

- (c) The following bubble sort pseudocode algorithm sorts the data in `theArray` into descending numerical order. There are **five** incomplete statements.

```
PROCEDURE bubbleSort()

    DECLARE temp : INTEGER

    FOR x ← 0 to .....
        FOR y ← 0 to .....
            IF theArray[y] ..... theArray[y + 1] THEN
                temp ← theArray[y]
                theArray[y] ← .....
                theArray[y + 1] ← .....
            ENDIF
        NEXT y
    NEXT x
ENDPROCEDURE
```

Write program code for the procedure `bubbleSort()` to sort the data in `arrayData` into descending order.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[6]

- 3 A computer game requires users to travel around a world to find and open treasure chests. Each treasure chest has a mathematics question inside. The user enters the answer. The number of points awarded depends on the number of attempts before the user gives the correct answer.

The program will be created using object-oriented programming (OOP).

The following class diagram describes the class `TreasureChest`.

TreasureChest	
question : STRING answer : INTEGER points : INTEGER	// stores the question // stores the answer // stores the maximum possible number of points available for this chest
constructor()	// takes question, answer and points as parameters and creates an instance of an object
getQuestion()	// returns the question
checkAnswer()	// takes the user's answer as a parameter and returns True if it is correct, otherwise returns False
getPoints()	// takes the number of attempts as a parameter and returns the number of points awarded

- (a) Create a new program.

Write program code to declare the class `TreasureChest`.

Do **not** write any other methods.

The attributes are private.

If you are using the Python programming language, include attribute declarations using comments.

Save your program as **question3**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[5]

- (b) The text file `TreasureChestData.txt` stores data for five questions, in the order of question, answer, points.

For example, the first three lines of the file are for the first question:

```
2*2 question  
4 answer  
10 points
```

Write program code for the procedure, `readData()` to:

- read each question, answer and points from the text file
- create an object of type `TreasureChest` for each question
- declare an array named `arrayTreasure` of type `TreasureChest`
- append each object to the array
- use exception handling to output an appropriate message if the file is not found.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[8]

- (c) The main program repeats each question until the user inputs the correct answer. The number of points awarded depends on the number of attempts before the user gives the correct answer.

- (i) The class `TreasureChest` has a method `getQuestion()` that returns the question.

Write the method `getQuestion()`.

Save your program.

Copy and paste the program code into **part 3(c)(i)** in the evidence document.

[1]

- (ii) The class `TreasureChest` has a method `checkAnswer()` that takes the user's answer as a parameter. It returns `True` if the answer is correct and `False` otherwise.

Write the method `checkAnswer()`.

Save your program.

Copy and paste the program code into **part 3(c)(ii)** in the evidence document.

[3]

- (iii) The class `TreasureChest` has a method `getPoints()` that takes the number of attempts as a parameter.

- If the number of attempts is 1, it returns the value of `points`.
- If the number of attempts is 2, it returns the integer value of `points` divided by 2 (`DIV 2`).
- If the number of attempts is 3 or 4, it returns the integer value of `points` divided by 4 (`DIV 4`).
- If the number of attempts is not 1 or 2 or 3 or 4, it returns 0 (zero).

For example, a question is worth 100 points and the user took 2 attempts to give the correct answer. The user is awarded 50 points ($100 \text{ DIV } 2$).

Write the method `getPoints()`.

Save your program.

Copy and paste the program code into **part 3(c)(iii)** in the evidence document.

[5]

(iv) Write program code for the main program to:

- call the procedure `readData()`
- ask the user to enter a question number between 1 and 5
- output the question that matches the question number entered by the user
- check if the answer input by the user is correct using the method `checkAnswer()`
- repeat the question until the user inputs the correct answer
- count how many times the user attempted the question
- use the method `getPoints()` to return the number of points awarded
- output the number of points the user is awarded.

Save your program.

Copy and paste the program code into **part 3(c)(iv)** in the evidence document.

[7]

(v) Test the program.

Take a screenshot showing the input(s) and output(s) for each of the following two tests.

In the first test:

- select question 1 and answer it correctly the first time.

In the second test:

- select question 5 and answer it correctly the second time.

Save your program.

Copy and paste the screenshots into **part 3(c)(v)** in the evidence document.

[2]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

October/November 2015

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

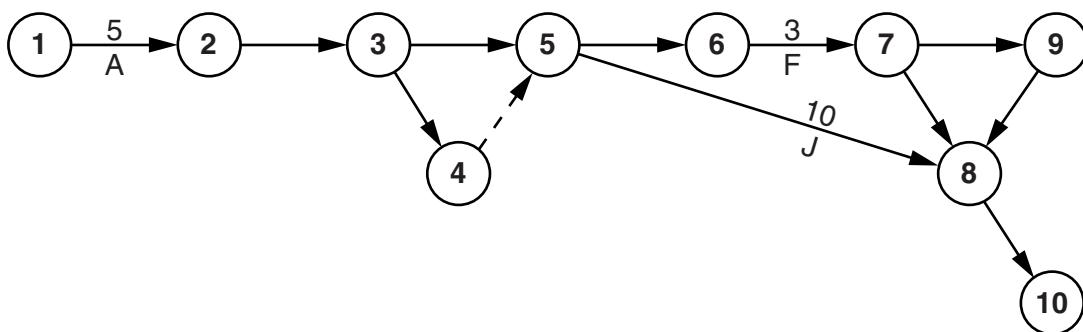
Complete the statement to indicate which high-level programming language you will use.

Programming language

- 1 A large software house has been asked to supply a computerised solution for a business. The project manager has drawn up a list of activities and their likely duration.

Activity	Description	Weeks to complete
A	Write requirement specification	5
B	Produce program design	5
C	Write module code	15
D	Module testing	10
E	Integration testing	5
F	Alpha testing	3
G	Install software and acceptance testing	5
H	Write end user training guide	5
J	Write technical documentation	10
K	End user training	4
L	Sign off final system	1

- (a) The project manager decides to construct a Program Evaluation Review Technique (PERT) chart from this data.



- (i) Complete the PERT chart.

[7]

- (ii) State the critical path.

..... [2]

- (iii) Calculate the minimum number of weeks for the completion of this solution.

..... [1]

(b) For activity J:

(i) State the earliest start time.

Week number [1]

(ii) State the latest start time.

Week number [1]

(c) Give a reason why the project manager used a PERT chart.

.....
..... [1]

- 2** A declarative programming language is used to represent the following facts and rules:

```

01 male(ali).
02 male(raul).
03 male(ahmed).
04 male(phiippe).
05 female(meena).
06 female(aisha).
07 female(gina).
08 parent(ali, raul).
09 parent(meena, raul).
10 parent(ali, ahmed).
11 parent(meena, ahmed).
12 parent(ali, aisha).
13 parent(meena, aisha).
14 father(A, B) IF male(A) AND parent(A, B).

```

These clauses have the following meaning:

Clause	Explanation
01	Ali is male
05	Meena is female
08	Ali is a parent of Raul
14	A is the father of B if A is male and A is a parent of B

- (a)** More facts are to be included.

Philippe and Gina are the parents of Meena.

Write the additional clauses to record this.

15

16, [2]

- (b)** Using the variable P, the goal

parent(P, raul)

returns

P = ali, meena

Write the result returned by the goal

parent(ali, C)

C =, [2]

- (c) Use the variable F to write the goal to find the father of Ahmed.

..... [1]

- (d) Write the rule to show that X is the mother of Y .

$\text{mother}(X, Y)$

IF

..... [2]

- (e) W is a grandparent of Z if W is a parent of one of Z 's parents.

Complete the following rule:

$\text{grandparent}(W, Z)$

IF

..... [2]

- (f) Complete the rule to show that G is a grandfather of K .

$\text{grandfather}(G, K)$

IF

..... [2]

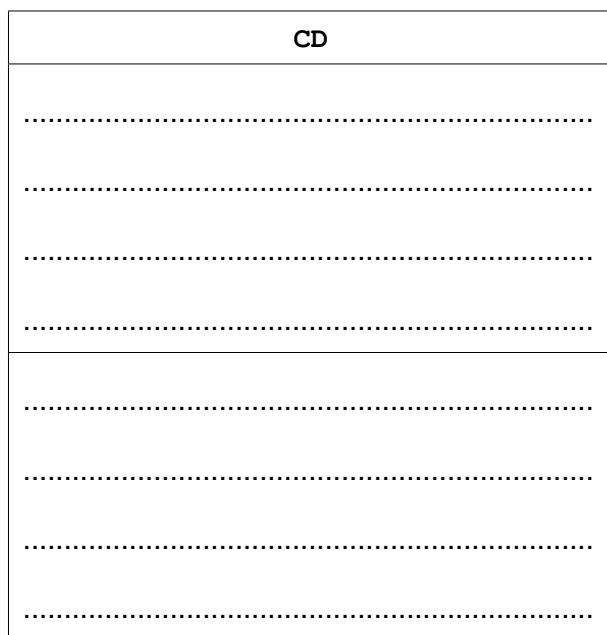
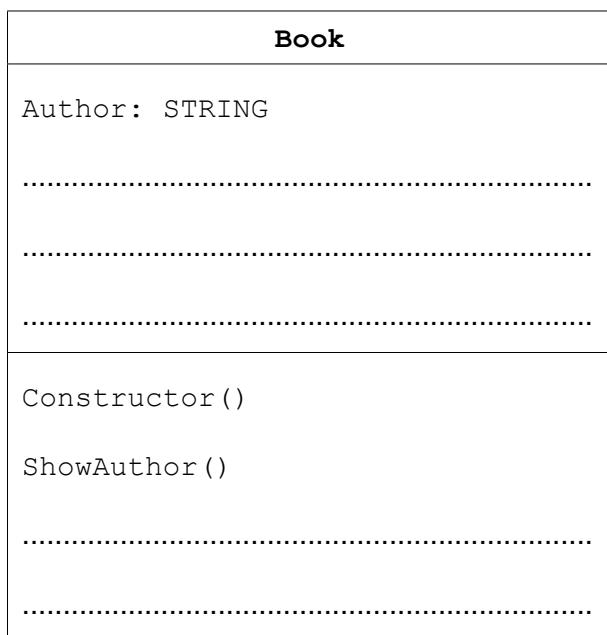
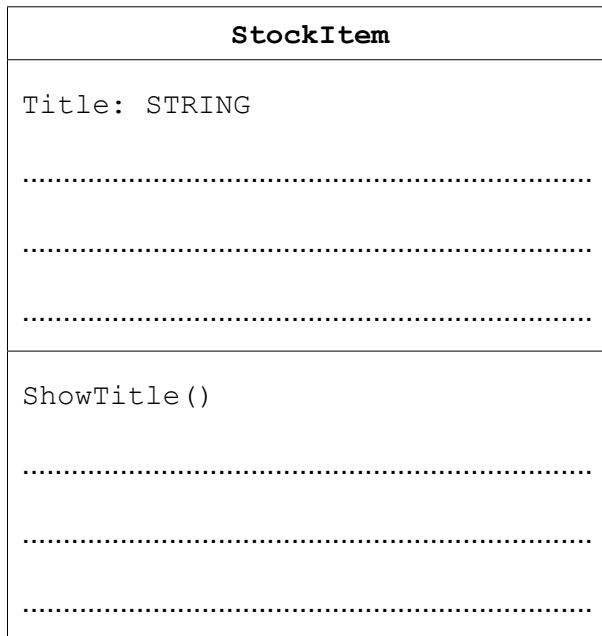
- 3 A lending library stocks two types of item for loan: books and CDs.

All stock items have a title, the date the item was acquired and whether the item is currently out on loan.

Books have an author and ISBN. CDs have an artist and play time in minutes.

The library needs a program to process data about the stock items. The program will use an object-oriented programming language.

- (a) Complete the class diagram showing the appropriate properties and methods.



(b) Write program code

- (i) for the class definition for the superclass StockItem.

- (ii) for the class definition for the subclass Book.

(iii) to create a new instance of Book with:

- identifier NewBook
- title "Computers"
- author A.Nyone
- ISBN 099111
- acquired on 12/11/2001
- not out on loan

Programming language
.....
.....
.....
.....
.....
.....
.....
..... [3]

Question 4 begins on page 10.

4 A binary tree Abstract Data Type (ADT) has these associated operations:

- create the tree (CreateTree)
- add an item to tree (Add)
- output items in ascending order (TraverseTree)

(a) Show the final state of the binary tree after the following operations are carried out.

```
CreateTree  
Add("Dodi")  
Add("Farai")  
Add("Elli")  
Add("George")  
Add("Ben")  
Add("Celine")  
Add("Ali")
```

[4]

- (b) The binary tree ADT is to be implemented as an array of nodes. Each node consists of data and two pointers.

Using pseudocode, a record type, Node, is declared as follows:

```
TYPE Node
    DECLARE Name : STRING
    DECLARE LeftPointer : INTEGER
    DECLARE RightPointer : INTEGER
ENDTYPE
```

The statement

```
DECLARE Tree : ARRAY[1:10] OF Node
```

reserves space for 10 nodes in array Tree.

The CreateTree operation links all nodes into a linked list of free nodes. It also initialises the RootPointer and FreePointer.

Show the contents of the Tree array and the values of the two pointers, RootPointer and FreePointer, after the operations given in part (a) have been carried out.

Tree			
RootPointer	Name	LeftPointer	RightPointer
[1]			
[2]			
[3]			
[4]			
[5]			
[6]			
[7]			
[8]			
[9]			
[10]			

[7]

- (c) A programmer needs an algorithm for outputting items in ascending order. To design this, the programmer writes a recursive procedure in pseudocode.

- (i) Complete the pseudocode:

```

01 PROCEDURE TraverseTree (BYVALUE Root: INTEGER)

02     IF Tree[Root].LeftPointer ......

03         THEN

04             TraverseTree (.....)

05         ENDIF

06         OUTPUT ..... Name

07         IF ..... <> 0

08             THEN

09                 TraverseTree (.....)

10            ENDIF

11 ENDPROCEDURE

```

[5]

- (ii) Explain what is meant by a recursive procedure. Give a line number from the code above that shows procedure `TraverseTree` is recursive.

.....
.....
.....

Line number [2]

- (iii) Write the pseudocode call required to output all names stored in `Tree`.

.....
.....

[1]

Question 5 begins on page 14.

- 5 Data about sports club members are stored in a random file of records.

- The key field of a member record is the member ID (range 1000 to 9999).
- Other member data are stored.
- A hashing function is used to calculate a record address.
- The random file initially consists of dummy records.
- Dummy records are shown by member ID set to 0.

```
FUNCTION Hash(MemberID : INTEGER) RETURNS INTEGER

    Address ← MemberID MOD 100

    RETURN Address

ENDFUNCTION
```

- (a) New members with the following member IDs have joined the sports club:

1001, 3005, 4096, 2098, 7002

Indicate where each record should be stored by deleting the zero and writing the member ID in the correct cell.

MembershipFile		
Address	MemberID	Other member data
0	0	
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
:		
:		
96	0	
97	0	
98	0	
99	0	

[2]

- (b) (i)** The program stores a new member's data in the record variable `NewMember`. The field `MemberID` stores the member ID.

Complete the pseudocode:

```

10 // generate record address
20 NewAddress ← .....
30 // move pointer to the disk address for the record
40 SEEK .....
50 PUTRECORD "MembershipFile", .....

```

[4]

- (ii)** Before records can be saved to the file `MembershipFile`, the file needs to be opened.

Complete the pseudocode.

```

01 TRY
02      OPENFILE ..... FOR RANDOM
03 EXCEPT
04      .....
05 ENDTRY

```

[2]

- (iii)** A record with member ID 9001 is to be stored.

Explain the problem that occurs when this record is saved.

.....
.....
.....
.....

[2]

- (iv)** Describe a method, without changing the function `Hash`, to handle the problem identified in part (b)(iii).

.....
.....
.....
.....

[2]

- (v) Write **pseudocode** to implement the method you described in part (b)(iv).

Choose line numbers to indicate where your pseudocode should be inserted in the pseudocode of **part (b)(i)**.

. [4]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

May/June 2019

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

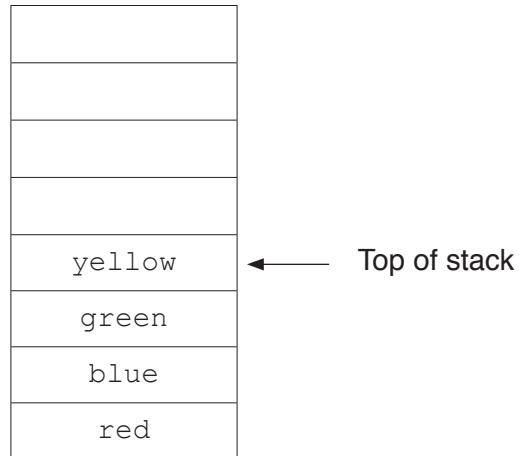
At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **18** printed pages and **2** blank pages.

- 1 (a) A stack contains the values 'red', 'blue', 'green' and 'yellow'.



- (i) Show the contents of the stack in **part(a)** after the following operations.

POP()

PUSH('purple')

PUSH('orange')



[1]

- (ii) Show the contents of the stack from **part(a)(i)** after these further operations.

POP ()

POP ()

PUSH ('brown')

POP ()

PUSH ('black')



[1]

- (b) A queue is an alternative Abstract Data Type (ADT).

Describe a **queue**.

.....
.....
.....
.....
.....
.....
.....
.....

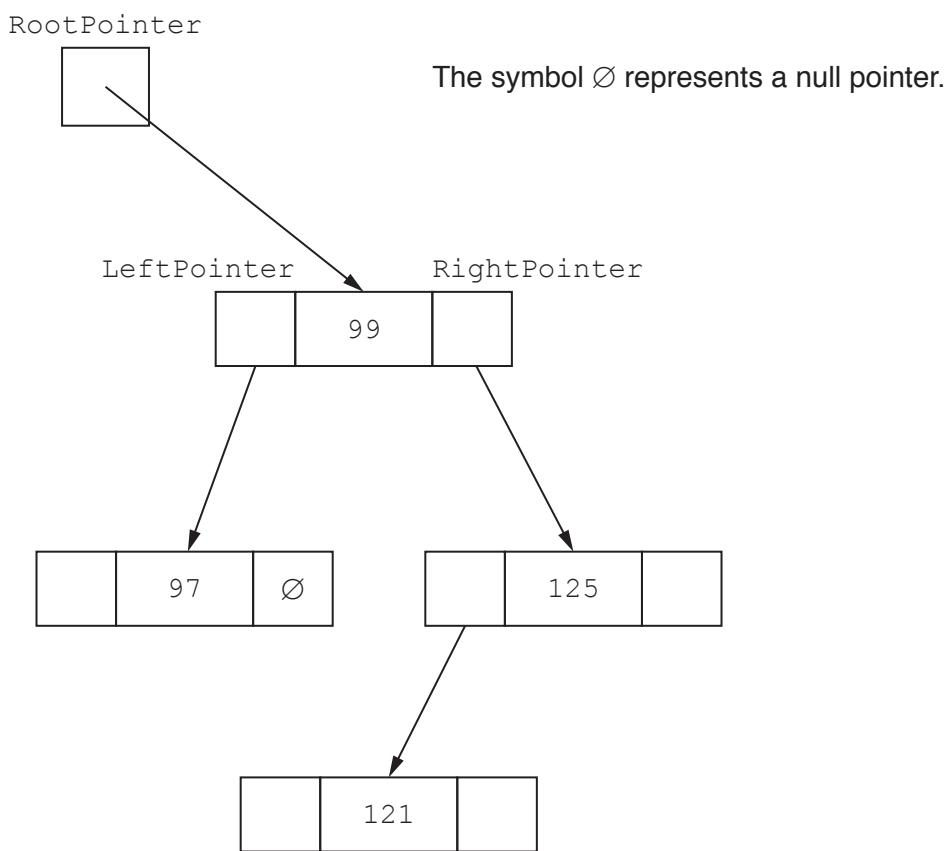
[3]

- 2 A computer games club wants to run a competition. The club needs a system to store the scores achieved in the competition.

A selection of score data is as follows:

99, 125, 121, 97, 109, 95, 135, 149

- (a) A linked list of nodes will be used to store the data. Each node consists of the data, a left pointer and a right pointer. The linked list will be organised as a binary tree.
- (i) Complete the binary tree to show how the score data above will be organised.



[5]

- (ii) The following diagram shows a 2D array that stores the nodes of the binary tree's linked list.

Add the correct pointer values to complete the diagram, using your answer from part (a)(i).

RootPointer	Index	LeftPointer	Data	RightPointer
<input type="text"/>	0		99	
	1		125	
	2		121	
FreePointer	3		97	
<input type="text"/>	4		109	
	5		95	
	6		135	
	7		149	
	8			

[6]

- (b) The club also considers storing the data in the order in which it receives the scores as a linked list in a 1D array of records.

The following pseudocode algorithm searches for an element in the linked list.

Complete the **six** missing sections in the algorithm.

```
FUNCTION FindElement(Item : INTEGER) RETURNS .....  
..... ← RootPointer  
WHILE CurrentPointer ..... NullPointer  
IF List[CurrentPointer].Data <> .....  
THEN  
    CurrentPointer ← List[.....] . Pointer  
ELSE  
    RETURN CurrentPointer  
ENDIF  
ENDWHILE  
CurrentPointer ← NullPointer  
..... CurrentPointer  
ENDFUNCTION
```

[6]

- (c) The games club is looking at two programming paradigms: imperative and object-oriented programming paradigms.

Describe what is meant by the **imperative programming paradigm** and the **object-oriented programming paradigm**.

(i) Imperative

.....
.....
.....
.....
..... [3]

(ii) Object-oriented

.....
.....
.....
.....
..... [3]

- (d) Players complete one game to place them into a category for the competition. The games club wants to implement a program to place players into the correct category. The programmer has decided to use object-oriented programming (OOP).

The highest score that can be achieved in the game is 150. Any score less than 50 will not qualify for the competition. Players will be placed in a category based on their score.

The following diagram shows the design for the class `Player`. This includes the properties and methods.

Player	
<code>Score : INTEGER</code>	// initialised to 0
<code>Category : STRING</code>	// "Beginner", "Intermediate", // "Advanced" or "Not Qualified", initialised // to "Not Qualified"
<code>PlayerID : STRING</code>	// initialised with the parameter InputPlayerID
<code>Create()</code>	// method to create and initialise an object using // language-appropriate constructor
<code>SetScore()</code>	// checks that the Score parameter has a valid value // if so, assigns it to Score
<code>SetCategory()</code>	// sets Category based on player's Score
<code>SetPlayerID()</code>	// allows a player to change their PlayerID // validates the new PlayerID
<code>GetScore()</code>	// returns Score
<code>GetCategory()</code>	// returns Category
<code>GetPlayerID()</code>	// returns PlayerID

- (i) The constructor receives the parameter `InputPlayerID` to create the `PlayerID`. Other properties are initialised as instructed in the class diagram.

Write program code for the Create() constructor method.

Programming language

Program code

[5]

[5]

- (ii) Write **program code** for the following **three** get methods.

Programming language

GetScore ()

Program code

.....
.....
.....
.....

GetCategory ()

Program code

.....
.....
.....
.....

GetPlayerID ()

Program code

.....
.....
.....
.....

[4]

- (iii) The method `SetPlayerID()` asks the user to input the new player ID and reads in this value.

It checks that the length of the PlayerID is less than or equal to 15 characters and greater than or equal to 4 characters. If the input is valid, it sets this as the PlayerID, otherwise it loops until the player inputs a valid PlayerID.

Use suitable input and output messages.

Write program code for SetPlayerID().

Programming language

Program code

[4]

- (iv) The method `SetScore()` checks that its `INTEGER` parameter `ScoreInput` is valid. If it is valid, it is then set as `Score`. A valid `ScoreInput` is greater than or equal to 0 and less than or equal to 150.

If the ScoreInput is valid, the method sets Score and returns TRUE.

If the `ScoreInput` is not valid, the method does not set `Score`, displays an error message, and it returns `FALSE`.

Write program code for SetScore (ScoreInput : INTEGER).

Programming language

Program code

[5]

[5]

- (v) Write **program code** for the method `SetCategory()`. Use the properties and methods in the original class definition.

Players will be placed in one of the following categories.

Category	Criteria
Advanced	Score is greater than 120
Intermediate	Score is greater than 80 and less than or equal to 120
Beginner	Score is greater than or equal to 50 and less than or equal to 80
Not Qualified	Score is less than 50

Programming language

Program code

[4]

- (vi) Joanne has played the first game to place her in a category for the competition.

The procedure `CreatePlayer()` performs the following tasks.

- allows the player ID and score to be input with suitable prompts
 - creates an instance of `Player` with the identifier `JoannePlayer`
 - sets the score for the object
 - sets the category for the object
 - outputs the category for the object

Write program code for the CreatePlayer() procedure.

Programming language

Program code

[8]

[8]

- (e) The programmer wants to test that the correct category is set for a player's score.

As stated in **part (d)(v)**, players will be placed in one of the following categories.

Category	Criteria
Advanced	Score is greater than 120
Intermediate	Score is greater than 80 and less than or equal to 120
Beginner	Score is greater than or equal to 50 and less than or equal to 80
Not Qualified	Score is less than 50

Complete the table to provide test data for each category.

Category	Type of test data	Example test data
Beginner	Normal	
	Abnormal	
	Boundary	
Intermediate	Normal	
	Abnormal	
	Boundary	
Advanced	Normal	
	Abnormal	
	Boundary	

[3]

- (f) In part (b), the club stored scores in a 1D array. This allows the club to sort the scores.

The following is a sorting algorithm in pseudocode.

```
NumberOfScores ← 5  
  
FOR Item ← 1 TO NumberOfScores - 1  
  
    InsertScore ← ArrayData[Item]  
  
    Index ← Item - 1  
  
    WHILE (ArrayData[Index] > InsertScore) AND (Index ≥ 0)  
        ArrayData[Index + 1] ← ArrayData[Index]  
  
        Index ← Index - 1  
  
    ENDWHILE  
  
    ArrayData[Index + 1] ← InsertScore  
  
ENDFOR
```

- (i) Give the name of this algorithm.

..... [1]

- (ii) State the name of **one** other sorting algorithm.

..... [1]

- (iii) Complete a dry run of the algorithm using the following trace table.

[7]

- 3 Some algorithms can be written using recursion.

- (a) State **two** features of recursion.

Feature 1

Feature 2

[2]

- (b) Explain what a compiler has to do to implement recursion.

.....
.....
.....
.....
.....
..... [3]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/41

Paper 4 Practical

May/June 2023

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the evidence document **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

One source file is used to answer **Question 1** and two source files are used to answer **Question 3**. The files are called **Data.txt**, **AnimalData.txt** and **ColourData.txt**

A class declaration can be used to declare a record.

A list is an alternative to an array.

1 A program reads data from a file and searches for specific data.

- (a) The main program needs to read 25 integer data items from the text file **Data.txt** into a local 1D array, **DataArray**
 - (i) Write program code to declare the local array **DataArray**

Save your program as **Question1_J2023**.

Copy and paste the program code into **part 1(a)(i)** in the evidence document.

[1]

- (ii) Amend the main program to read the contents of **Data.txt** into **DataArray**

Save your program.

Copy and paste the program code into **part 1(a)(ii)** in the evidence document.

[4]

- (b) (i) The procedure **PrintArray()** takes an integer array as a parameter and outputs the contents of the array in the order they are stored.

The items are printed on the same line, for example:

10 4 5 13 25

Write program code for the procedure **PrintArray()**

Save your program.

Copy and paste the program code into **part 1(b)(i)** in the evidence document.

[3]

- (ii) Amend the main program to output the contents of `DataArray` using the procedure `PrintArray()`

Save your program.

Copy and paste the program code into **part 1(b)(ii)** in the evidence document.

[1]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 1(b)(iii)** in the evidence document.

[1]

- (c) The function `LinearSearch()`:

- takes an integer array and integer search value as parameters
- counts and returns the number of times the search value is found in the array.

Write program code for the function `LinearSearch()`

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[3]

- (d) (i) Amend the main program to:

- prompt the user to input a whole number between 0 and 100 inclusive
- read and validate the input from the user
- call `LinearSearch()` with `DataArray` and the validated input value
- output the result in the format:

The number 7 is found 2 times.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[4]

- (ii) Test your program by inputting the number 12.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 1(d)(ii)** in the evidence document.

[1]

- 2 A computer game is being designed that will include different vehicles. A prototype for the game is being developed using object-oriented programming.

The class `Vehicle` stores data about the vehicles. Each vehicle has an identification name, a maximum speed, a current speed and a horizontal position. The value `IncreaseAmount` is added to the current speed each time the vehicle increases its speed.

Vehicle	
<code>ID : STRING</code>	stores the identification name for the vehicle
<code>MaxSpeed : INTEGER</code>	stores the maximum speed
<code>CurrentSpeed : INTEGER</code>	stores the current speed
<code>IncreaseAmount : INTEGER</code>	stores the amount <code>CurrentSpeed</code> increases by
<code>HorizontalPosition : INTEGER</code>	stores the horizontal position
<code>Constructor()</code>	initialises <code>ID</code> , <code>MaxSpeed</code> and <code>IncreaseAmount</code> to the parameter values initialises both <code>CurrentSpeed</code> and <code>HorizontalPosition</code> to 0
<code>GetCurrentSpeed()</code>	returns the current speed
<code>GetIncreaseAmount()</code>	returns the increase amount
<code>GetHorizontalPosition()</code>	returns the horizontal position
<code>GetMaxSpeed()</code>	returns the maximum speed
<code>SetCurrentSpeed()</code>	assigns the parameter to the current speed
<code>SetHorizontalPosition()</code>	assigns the parameter to the horizontal position
<code>IncreaseSpeed()</code>	calculates and stores the new speed and horizontal position of the vehicle

- (a) (i) Write program code to declare the class `Vehicle`. All attributes must be private.

You only need to declare the class and its constructor. Do not declare any other methods.

Use your programming language's appropriate constructor.

If you are writing program code in Python, include attribute declarations using comments.

Save your program as **Question2_J2023**.

Copy and paste the program code into **part 2(a)(i)** in the evidence document.

[5]

- (ii) Write program code for the get methods `GetCurrentSpeed()`, `GetIncreaseAmount()`, `GetMaxSpeed()` and `GetHorizontalPosition()`

Save your program.

Copy and paste the program code into **part 2(a)(ii)** in the evidence document.

[3]

- (iii) Write program code for the set methods `SetCurrentSpeed()` and `SetHorizontalPosition()`

Save your program.

Copy and paste the program code into **part 2(a)(iii)** in the evidence document.

[3]

- (iv) The method `IncreaseSpeed()`:

- adds `IncreaseAmount` to the current speed
- adds the updated current speed to the horizontal position.

The current speed of a vehicle cannot exceed its maximum speed.

Write program code for the method `IncreaseSpeed()`

Save your program.

Copy and paste the program code into **part 2(a)(iv)** in the evidence document.

[3]

- (b) The child class `Helicopter` inherits from the parent class `Vehicle`. A helicopter also has a vertical position and changes the vertical position when it increases speed.

Helicopter	
<code>VerticalPosition : INTEGER</code>	stores the vertical position
<code>VerticalChange : INTEGER</code>	stores the amount <code>VerticalPosition</code> changes by
<code>MaxHeight : INTEGER</code>	stores the maximum height the helicopter can reach
<code>Constructor()</code>	takes the ID, maximum speed, increase amount, vertical change and maximum height as parameters initialises the vertical position to 0
<code>GetVerticalPosition()</code>	returns the vertical position
<code>IncreaseSpeed()</code>	changes the current speed, horizontal and vertical position of the helicopter

- (i) Write program code to declare the class `Helicopter`. You only need to declare the class and its constructor. You do not need to declare the other methods.

Use your programming language's appropriate constructor.

All attributes must be private.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into **part 2(b)(i)** in the evidence document.

[5]

- (ii) The `Helicopter` method `IncreaseSpeed()` overrides the method from the parent class and:

- adds the amount of vertical change to the vertical position
- adds `IncreaseAmount` to the current speed
- adds the updated current speed to the horizontal position.

The vertical position of a helicopter cannot exceed its maximum height.

The current speed of a helicopter cannot exceed its maximum speed.

Write program code for the method `IncreaseSpeed()`

Save your program.

Copy and paste the program code into **part 2(b)(ii)** in the evidence document.

[4]

- (c) A procedure needs to output the horizontal position and speed of a vehicle. If the vehicle is a helicopter, it also outputs the vertical position.

All outputs must include appropriate messages.

Write program code for this procedure.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[3]

- (d) The main program needs to:

- instantiate a car as a new vehicle with the ID "Tiger", a maximum speed of 100 and an increase amount of 20
- instantiate a new helicopter with the ID "Lion", a maximum speed of 350, an increase amount of 40, a vertical change of 3 and a maximum height of 100
- call `IncreaseSpeed()` twice for the car and then call the output procedure from **part 2(c)** for the car
- call `IncreaseSpeed()` twice for the helicopter and then call the output procedure from **part 2(c)** for the helicopter.

- (i) Write program code for the main program.

Save your program.

Copy and paste the program code into **part 2(d)(i)** in the evidence document.

[5]

- (ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 2(d)(ii)** in the evidence document.

[1]

- 3 A program implements two stacks using 1D arrays. One stack stores the names of colours. One stack stores the names of animals.

(a) The program contains the following global arrays and variables:

- 1D array `Animal` to store the names of up to 20 animals.
- 1D array `Colour` to store the names of up to 10 colours.
- `AnimalTopPointer` to point to the next free space in the array `Animal`, initialised to 0.
- `ColourTopPointer` to point to the next free space in the array `Colour`, initialised to 0.

Write program code to declare the global arrays and variables.

Save your program as **Question3_J2023**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[3]

(b) (i) Study the pseudocode function `PushAnimal()`:

```

FUNCTION PushAnimal(DataToPush : STRING) RETURNS BOOLEAN

IF AnimalTopPointer = 20 THEN
    RETURN FALSE
ELSE
    Animal[AnimalTopPointer] ← DataToPush
    AnimalTopPointer ← AnimalTopPointer + 1
    RETURN TRUE
ENDIF

ENDFUNCTION

```

Write program code for the function `PushAnimal()`

Save your program.

Copy and paste the program code into **part 3(b)(i)** in the evidence document.

[3]

(ii) Study the pseudocode function `PopAnimal()`:

```

FUNCTION PopAnimal() RETURNS STRING

    DECLARE ReturnData : STRING

    IF AnimalTopPointer = 0 THEN

        RETURN ""

    ELSE

        ReturnData ← Animal[AnimalTopPointer - 1]

        AnimalTopPointer ← AnimalTopPointer - 1

        RETURN ReturnData

    ENDIF

ENDFUNCTION

```

Write program code to declare the function `PopAnimal()`

Save your program.

Copy and paste the program code into **part 3(b)(ii)** in the evidence document.

[3]

(iii) The procedure `ReadData()`:

- reads the animal names from the file `AnimalData.txt`
- uses `PushAnimal()` to insert each name onto the stack
- uses appropriate exception handling if the file does not exist.

Write program code for the procedure `ReadData()`

Save your program.

Copy and paste the program code into **part 3(b)(iii)** in the evidence document.

[5]

(iv) The function `PushColour()` performs the same actions as `PushAnimal()` but inserts an item into `Colour`.

The function `PopColour()` performs the same actions as `PopAnimal()` but removes the next item from `Colour`.

Write program code for the functions `PushColour()` and `PopColour()`

Save your program.

Copy and paste the program code into **part 3(b)(iv)** in the evidence document.

[2]

[Turn over

(v) Amend the procedure `ReadData()` so that it also:

- reads the colours from the text file `ColourData.txt`
- uses `PushColour()` to insert each colour onto the stack
- uses appropriate exception handling if the file does not exist.

Save your program.

Copy and paste the program code into **part 3(b)(v)** in the evidence document.

[2]

(c) The procedure `OutputItem()`:

- pops the next item from both `Animal` and `Colour`
- outputs the colour and animal on one line, for example "black horse"

If there is no data in `Colour`:

- the animal is pushed back onto `Animal`
- "No colour" is output.

If there is no data in `Animal`:

- the colour is pushed back onto `Colour`
- "No animal" is output.

Write program code for the procedure `OutputItem()`

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[5]

(d) The main program:

- calls the procedure `ReadData()`
- calls `OutputItem()` **four times**.

(i) Write program code for the main program.

Save your program.

Copy and paste the program code into **part 3(d)(i)** in the evidence document.

[1]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 3(d)(ii)** in the evidence document.

[1]

BLANK PAGE

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.

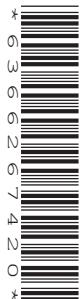
CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

May/June 2017

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **17** printed pages and **3** blank pages.

- 1 The following table shows part of the instruction set for a processor which has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
STO	<address>	Store the contents of ACC at the given address.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
CMP	<address>	Compare the contents of ACC with the contents of <address>.
JMP	<address>	Jump to the given address.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

- (a) A programmer writes a program that:

- reads two characters input from the keyboard (you may assume they will be capital letters in ascending alphabetical sequence)
- outputs the alphabetical sequence of characters from the first to the second character. For example, if the characters 'B' and 'F' are input, the output is:

BCDEF

The programmer has started to write the program in the following table. The Comment column contains descriptions for the missing program instructions, labels and data.

Complete the following program. Use op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// INPUT character
			// store in CHAR1
			// INPUT character
			// store in CHAR2
			// initialise ACC to ASCII value of CHAR1
			// output contents of ACC
			// compare ACC with CHAR2
			// if equal jump to end of FOR loop
			// increment ACC
			// jump to LOOP
ENDFOR:	END		
CHAR1:			
CHAR2:			

[9]

(b) The programmer now starts to write a program that:

- converts a positive integer, stored at address NUMBER1, into its negative equivalent in two's complement form
- stores the result at address NUMBER2

Complete the following program. Use op codes from the given instruction set.
Show the value stored in NUMBER2.

Label	Op code	Operand	Comment
START:			
		MASK	// convert to one's complement
			// convert to two's complement
	END		
MASK:			// show value of mask in binary here
NUMBER1:	B00000101		// positive integer
NUMBER2:			// negative equivalent

[6]

2 An ordered binary tree Abstract Data Type (ADT) has these associated operations:

- create tree
- add new item to tree
- traverse tree

The binary tree ADT is to be implemented as a linked list of nodes.

Each node consists of data, a left pointer and a right pointer.

(a) A null pointer is shown as \emptyset .

Explain the meaning of the term **null pointer**.

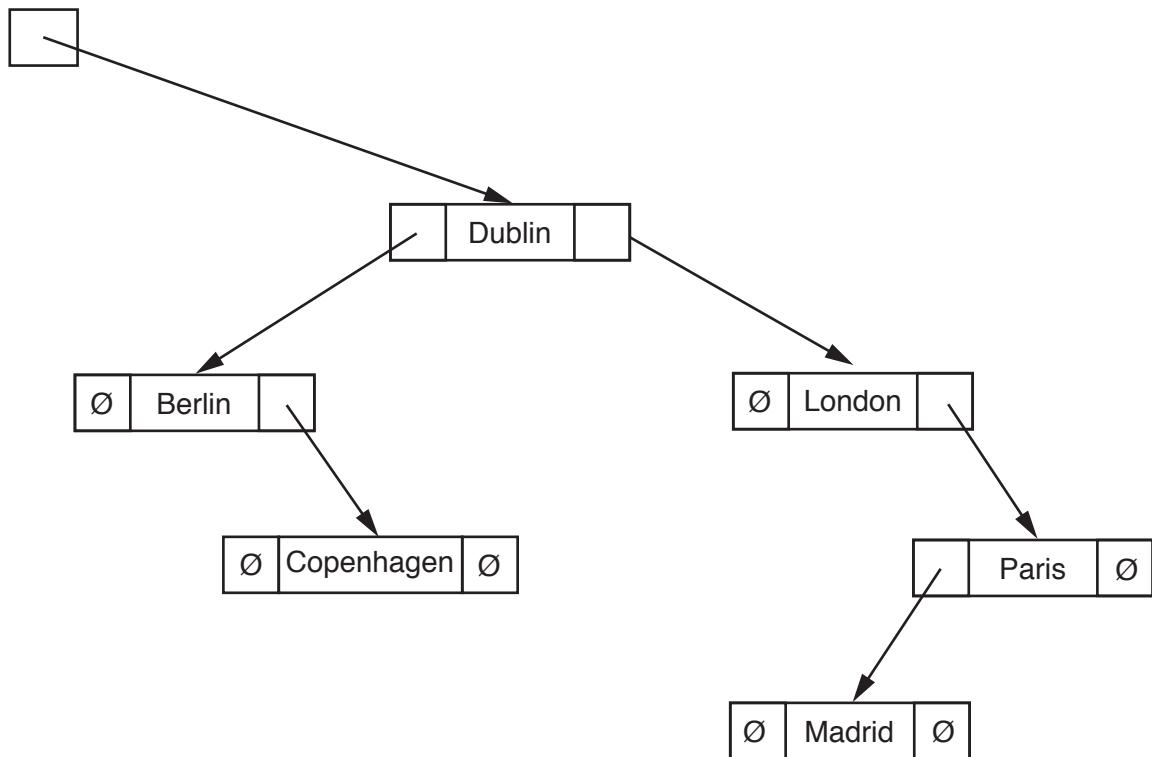
.....
.....

[1]

(b) The following diagram shows an ordered binary tree after the following data have been added:

Dublin, London, Berlin, Paris, Madrid, Copenhagen

RootPointer



Another data item to be added is Athens.

Make the required changes to the diagram when this data item is added.

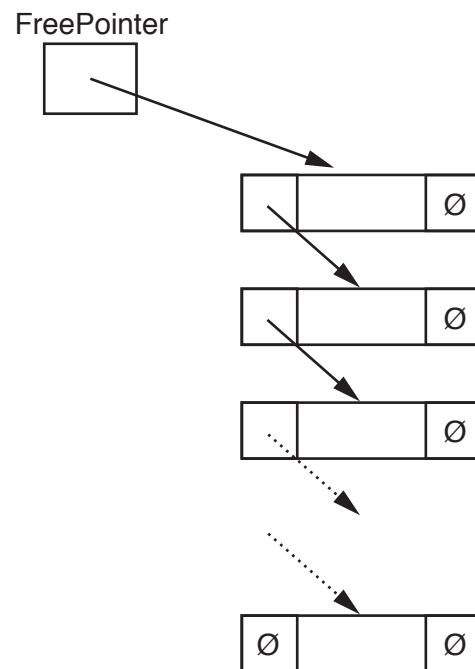
[2]

- (c) A tree without any nodes is represented as:

RootPointer



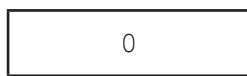
Unused nodes are linked together into a free list as shown:



The following diagram shows an array of records that stores the tree shown in part (b).

- (i) Add the relevant pointer values to complete the diagram.

RootPointer



FreePointer



	LeftPointer	Tree data	RightPointer
[0]		Dublin	
[1]		London	
[2]		Berlin	
[3]		Paris	
[4]		Madrid	
[5]		Copenhagen	
[6]		Athens	
[7]			
[8]			
[9]			

[5]

- (ii) Give an appropriate numerical value to represent the null pointer for this design. Justify your answer.
-
.....
.....
.....

[2]

- (d) A program is to be written to implement the tree ADT. The variables and procedures to be used are listed below:

Identifier	Data type	Description
Node	RECORD	Data structure to store node data and associated pointers.
LeftPointer	INTEGER	Stores index of start of left subtree.
RightPointer	INTEGER	Stores index of start of right subtree.
Data	STRING	Data item stored in node.
Tree	ARRAY	Array to store nodes.
NewDataItem	STRING	Stores data to be added.
FreePointer	INTEGER	Stores index of start of free list.
RootPointer	INTEGER	Stores index of root node.
NewNodePointer	INTEGER	Stores index of node to be added.
CreateTree()		Procedure initialises the root pointer and free pointer and links all nodes together into the free list.
AddToTree()		Procedure to add a new data item in the correct position in the binary tree.
FindInsertionPoint()		Procedure that finds the node where a new node is to be added. Procedure takes the parameter NewDataItem and returns two parameters: <ul style="list-style-type: none"> • Index, whose value is the index of the node where the new node is to be added • Direction, whose value is the direction of the pointer ("Left" or "Right").

- (i) Complete the pseudocode to create an empty tree.

```
TYPE Node  
.....  
.....  
.....  
ENDTYPE  
  
DECLARE Tree : ARRAY[0 : 9] .....,  
FreePointer : INTEGER,  
RootPointer : INTEGER  
  
PROCEDURE CreateTree()  
    DECLARE Index : INTEGER  
    .....  
    .....  
    FOR Index ← 0 TO 9    // link nodes  
    .....  
    .....  
    ENDFOR  
    .....  
ENDPROCEDURE
```

[7]

(ii) Complete the pseudocode to add a data item to the tree.

```

PROCEDURE AddToTree (BYVALUE NewDataItem : STRING)
// if no free node report an error
IF FreePointer .....
THEN
    OUTPUT ("No free space left")
ELSE // add new data item to first node in the free list
    NewNodePointer ← FreePointer
    .....
// adjust free pointer
FreePointer ← .....
// clear left pointer
Tree [NewNodePointer].LeftPointer ← .....
// is tree currently empty ?
IF .....
THEN // make new node the root node
    .....
ELSE // find position where new node is to be added
    Index ← RootPointer
    CALL FindInsertionPoint (NewDataItem, Index, Direction)
    IF Direction = "Left"
        THEN // add new node on left
        .....
    ELSE // add new node on right
        .....
    ENDIF
ENDIF
ENDIF
ENDPROCEDURE

```

[8]

- (e) The traverse tree operation outputs the data items in alphabetical order. This can be written as a recursive solution.

Complete the pseudocode for the recursive procedure `TraverseTree`.

```
PROCEDURE TraverseTree (BYVALUE Pointer : INTEGER)
```

ENDPROCEDURE

[5]

- 3 A programmer is writing a treasure island game to be played on the computer. The island is a rectangular grid, 30 squares by 10 squares. Each square of the island is represented by an element in a 2D array. The top left square of the island is represented by the array element [0, 0]. There are 30 squares across and 10 squares down.

The computer will:

- generate three random locations where treasure will be buried
- prompt the player for the location of one square where the player chooses to dig
- display the contents of the array by outputting for each square:
 - ' .' for only sand in this square
 - ' T ' for treasure still hidden in sand
 - ' X ' for a hole dug where treasure was found
 - ' O ' for a hole dug where no treasure was found.

Here is an example display after the player has chosen to dig at location [9, 3]:

```
.....  
.....  
.....  
.....  
.....  
..... T ..  
.....  
.....  
..... T ..  
...X.....
```

The game is to be implemented using object-oriented programming.

The programmer has designed the class `IslandClass`. The identifier table for this class is:

Identifier	Data type	Description
Grid	ARRAY[0 : 9, 0 : 29] OF CHAR	2D array to represent the squares of the island
Constructor()		instantiates an object of class <code>IslandClass</code> and initialises all squares to sand
HideTreasure()		generates a pair of random numbers used as the grid location of treasure and marks the square with 'T'
DigHole(Row, Column)		takes as parameters a valid grid location and marks the square with 'X' or 'O' as appropriate
GetSquare(Row, Column)	CHAR	takes as parameter a valid grid location and returns the grid value for that square from the <code>IslandClass</code> object

- (a) The programmer designed the pseudocode for the main program as follows:

```
DECLARE Island : IslandClass.Constructor()      // instantiate object

CALL DisplayGrid()                            // output island squares

FOR Treasure ← 1 TO 3                         // hide 3 treasures

    CALL Island.HideTreasure()

ENDFOR

CALL StartDig()                               // user to input location of dig

CALL DisplayGrid()                            // output island squares
```

Write **program code** to implement this pseudocode.

Programming language used

Program code

[3]

. [3]

- (b) Write program code to declare the IslandClass and write the constructor method.

The value to represent sand should be declared as a constant.

Programming language used

Program code

. [5]

- (c) The procedure `DisplayGrid` shows the current grid data. `DisplayGrid` makes use of the getter method `GetSquare` of the `Island` class.

An example output is:

```
.....  
.....  
.....  
.....  
..... T ..  
.....  
..... T ..  
... X ..
```

- (i) Write **program code** for the `GetSquare (Row, Column)` getter method.

.....
.....
.....
.....
.....
..... [2]

- (ii) Write **program code** for the `DisplayGrid` procedure.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [4]

- (d) Write **program code** for the `HideTreasure` method. Your method should check that the random location generated does not already contain treasure.

The value to represent treasure should be declared as a constant.

Programming language used

Program code

[View Details](#) | [Edit](#) | [Delete](#)

[View Details](#) | [Edit](#) | [Delete](#)

.....

[View Details](#) | [Edit](#) | [Delete](#)

[View Details](#) | [Edit](#) | [Delete](#)

Digitized by srujanika@gmail.com

. [5]

- (e) (i) The `DigHole` method takes two integers as parameters. These parameters form a valid grid location. The location is marked with 'X' or 'O' as appropriate.

Write **program code** for the `DigHole` method. The values to represent treasure, found treasure and hole should be declared as constants.

Programming language used

Program code

[3]

. [3]

(ii) The StartDig procedure:

- prompts the player for a location to dig
 - validates the user input
 - calls the `DigHole` method from **part (e)(i)**.

Write **program code** for the StartDig procedure. Ensure that the user input is fully validated.

Programming language used

Program code

. [5]

- (f) (i) The squares in the IslandClass grid could have been declared as objects of a Square class.

State the term used to describe the relationship between IslandClass and Square.

.....

.....

[1]

- (ii) Draw the appropriate diagram to represent this relationship. Do not list the attributes and methods of the classes.

[2]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

October/November 2021

2 hours

You must answer on the question paper.

No additional materials are needed.

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.

This document has **20** pages. Any blank pages are indicated.

- 1 An array, `NumberArray`, stores 100 integer values. The array needs to be sorted into ascending numerical order.

(a) Describe how an insertion sort will sort the data in `NumberArray`.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [4]

- (b) Another type of sorting algorithm is a bubble sort.

The procedure Bubble() takes an array as a parameter. It performs a bubble sort on the array. The sorting algorithm stops as soon as all the elements are in ascending order.

Complete the procedure Bubble().

```

PROCEDURE Bubble(BYREF NumberArray : ARRAY[0 : 99] OF INTEGER)

    DECLARE Outer : INTEGER

    DECLARE Swap : BOOLEAN

    DECLARE Inner : INTEGER

    DECLARE Temp : INTEGER

    Outer ← LENGTH(NumberArray) - 1

    REPEAT

        Inner ← .....

        Swap ← FALSE

        REPEAT

            IF NumberArray[Inner] > NumberArray[Inner + 1]

                THEN

                    Temp ← NumberArray[Inner]

                    NumberArray[Inner] ← NumberArray[Inner + 1]

                    NumberArray[Inner + 1] ← Temp

                    Swap ← .....

            ENDIF

            Inner ← Inner + 1

        UNTIL Inner = .....

        Outer ← Outer - 1

    UNTIL Swap = ..... OR Outer = .....
```

ENDPROCEDURE

[5]

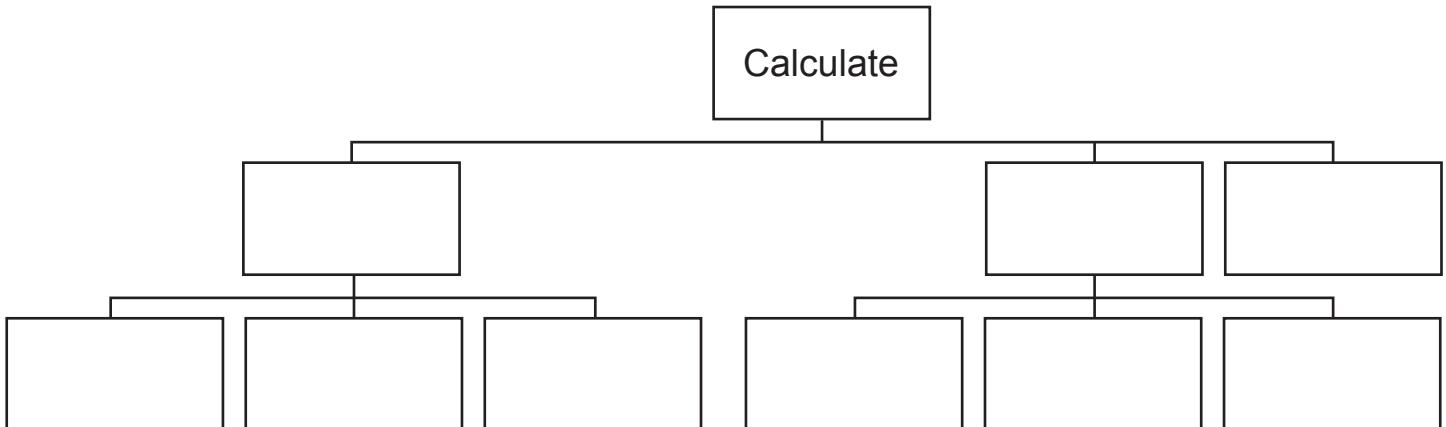
- 2 Complete the JSP structure diagram for the following pseudocode procedure.

```

PROCEDURE Calculate()
    INPUT Number1
    INPUT Number2
    INPUT Command
    IF Command = 1
        THEN
            Value ← Function1(Number1, Number2)
    ELSE
        IF Command = 2
            THEN
                Value ← Function2(Number1, Number2)
        ELSE
            Value ← Function3(Number1, Number2)
        ENDIF
    ENDIF
    OUTPUT Value
ENDPROCEDURE

```

JSP structure diagram



[4]

- 3 A user has to choose a new password to create an account. It is recommended that the password has at least two of the following elements:

- upper-case letter
- numeric character
- symbol.

The system outputs:

- "Strong" if there are at least two of the elements
- "Medium" if there is only one of the elements
- "Weak" if there are none of the elements.

Complete the following decision table for the password system described.

		Rules							
Conditions	One or more upper-case letters	N	Y	N	Y	N	Y	N	Y
	One or more numeric characters	N	N	Y	Y	N	N	Y	Y
	One or more symbols	N	N	N	N	Y	Y	Y	Y
Actions	Strong								
	Medium								
	Weak								

[3]

- 4 Each node of a binary tree is a record. Each record has a left pointer, an integer data value between 0 and 100 inclusive, and a right pointer.

For example:

Item	Example data
LeftPointer	2
Data	34
RightPointer	3

- (a) Write **pseudocode** to declare the record with the identifier Node.

.....
.....
.....
.....
.....
..... [2]

- (b) Write **pseudocode** to declare a new node, Node100, and assign 100 to its data value, 1 to the left pointer and 4 to the right pointer.

.....
.....
.....
.....
.....
.....
..... [3]

(c) The ordered binary tree is stored as a 1D global array named `BinaryTree` of type `Node`.

`RootNode` and `FreePointer` are declared as global variables.

A null pointer is represented by `-1`.

The current state of the binary tree is shown in the following table:

<code>RootNode</code>	0	<code>Index</code>	<code>LeftPointer</code>	<code>Data</code>	<code>RightPointer</code>
<code>FreePointer</code>	6	[0]	1	23	3
		[1]	-1	5	2
		[2]	-1	8	4
		[3]	5	100	-1
		[4]	-1	9	-1
		[5]	-1	88	-1
		[6]	-1	null	-1
		[7]	-1	null	-1

(i) State the purpose of the free pointer.

..... [1]

(ii) Identify an appropriate integer value to represent null data.

..... [1]

(iii) Draw the current state of the binary tree.

[2]

(iv) The procedure AddData ():

- takes the node to be added to the tree as a parameter
- finds the location for the node to be stored
- stores the node in the next free array index
- stores -1 in the new node's LeftPointer and RightPointer
- updates the pointers in the other nodes
- updates FreePointer.

Complete the pseudocode for the procedure AddData () .

```

PROCEDURE AddData (NewNode)

BinaryTree[FreePointer] ← .....
BinaryTree[FreePointer].LeftPointer ← -1
BinaryTree[FreePointer].RightPointer ← -1
DECLARE PositionFound : BOOLEAN
DECLARE PointerCounter : INTEGER
PositionFound ← .....
PointerCounter ← RootNode
WHILE NOT .....
  IF ..... < BinaryTree[PointerCounter].Data
    THEN
      IF BinaryTree[PointerCounter].LeftPointer = -1
        THEN
          BinaryTree[PointerCounter].LeftPointer ← FreePointer
          PositionFound ← TRUE
        ELSE
          PointerCounter ← BinaryTree[PointerCounter].LeftPointer
        ENDIF
      ELSE
        IF BinaryTree[PointerCounter].RightPointer = -1
          THEN
            BinaryTree[PointerCounter].RightPointer ← FreePointer
            PositionFound ← TRUE
          ELSE
            PointerCounter ← BinaryTree[PointerCounter].RightPointer
          ENDIF
        ENDIF
      ENDWHILE
      FreePointer ← FreePointer .....
ENDPROCEDURE

```

- 5 Study the following recursive pseudocode algorithm.

```

FUNCTION Recursive(Num1, Num2 : INTEGER) RETURNS INTEGER
    IF Num1 > Num2
        THEN
            RETURN 10
    ELSE
        IF Num1 = Num2
            THEN
                RETURN Num1
        ELSE
            RETURN Num1 + Recursive(Num1 * 2, Num2)
        ENDIF
    ENDIF
ENDFUNCTION

```

- (a) The function is called as follows:

Recursive(1, 15)

Dry run the function and complete the trace table. Give the final return value.

Trace table:

Function call	Num1	Num2	Return value

Final return value

Working

.....

.....

.....

.....

[4]

- (b) Rewrite the function Recursive() in **pseudocode**, using an **iterative algorithm**.

[7]

- 6 Details of errors generated in a program are stored in a stack.

Details of each error are stored in a record structure, Error.

- (a) State which error will be the first retrieved from the stack.

.....
..... [1]

- (b) The stack is implemented as a 1D array with the identifier ErrorArray.

The pointer LastItem stores the position of the last error in the array.

- (i) The function, AddItemToStack, takes the next error, the array, and pointer as parameters.

If the stack is full, the function returns FALSE; otherwise it adds the error to the stack, changes the pointer's value and returns TRUE.

Complete the following pseudocode for the function AddItemToStack.

```
FUNCTION AddItemToStack(BYREF ErrorArray : ARRAY[0 : 99] OF Error,
                      BYREF LastItem : INTEGER,
                      BYVALUE Error1 : Error) RETURNS BOOLEAN

  IF LastItem = .....
    THEN
      RETURN .....
    ELSE
      ErrorArray[LastItem + 1] ← .....
      LastItem ← .....
    RETURN .....
  ENDIF

ENDFUNCTION
```

- (ii) Explain the reasons why ErrorArray and LastItem are passed by reference, but Error1 is passed by value. [4]

.....
.....
.....
.....
..... [3]

(iii) The function RemoveItem takes the next error from the stack and returns it.

If there are no errors in the stack, it returns the global record NullError.

Complete the pseudocode algorithm RemoveItem.

```
FUNCTION RemoveItem(BYREF ErrorArray : ARRAY[0 : 99] OF Error,
                    BYREF LastItem : INTEGER) RETURNS Error

DECLARE ItemToRemove : Error

IF ..... THEN

    RETURN ......

ELSE

    ItemToRemove ← ErrorArray[.....]

    LastItem ← LastItem - 1

    RETURN ......

ENDFUNCTION
```

[3]

- (iv) The errors that have been processed are stored in a global queue, `ErrorComplete`.

The function `Enqueue` adds a record to `ErrorComplete`:

Enqueue (ErrorToAdd)

`Enqueue()` returns TRUE if the record is successfully added to the queue, and returns FALSE if the queue is full.

The procedure RunError() should:

- remove a record from the stack using the function `RemoveItem()`
 - output an appropriate message if there were no records in the stack
 - if an error record is returned, add the record to the queue using the function `Enqueue()`
 - if the record is added to the queue, output an appropriate message
 - if the record is not added to the queue, output an appropriate message.

Complete the pseudocode procedure RunError().

```
PROCEDURE RunError (BYREF ErrorComplete : ARRAY[0 : 99] OF Error,  
                    BYREF ErrorArray : ARRAY[0 : 99] OF Error)
```

ENDPROCEDURE

[5]

- 7 A treasure box is hidden within a computer game.

The box has a code that needs to be entered to allow the user into the box. The box contains up to 10 objects that are defined as being of the class `FieldObject`. The definition for the class `Box` is:

Box	
<code>Size : STRING</code>	// small, medium or large
<code>Contents : ARRAY[0 : 9] OF FieldObject</code>	// the 10 items the box holds
<code>Lock : STRING</code>	// the code to unlock the box
<code>Strength : INTEGER</code>	// the strength of the box // decreases by 1 each time an // incorrect code is entered
<code>Constructor()</code>	// instantiates an object of the Box // class and assigns initial values // to the attributes
<code>Unlock()</code>	// checks if the code is correct to // unlock the box
<code>GetContents()</code>	// returns the array
<code>SetSize()</code>	// sets the size of the box
<code>SetContents()</code>	// sets the contents of the box
<code>SetLock()</code>	// sets the lock code
<code>SetStrength()</code>	// sets the strength

- (a) The constructor creates a new instance of a box. It takes the size of the box, one item of content and the lock code as parameters. The strength is initialised to 100.

Write program code to create the class and constructor for Box.

Do not write the code for `Unlock()` or any of the set or get methods. Use the constructor for your chosen language.

Programming language

Program code

[5]

- (b) The player inputs the code to unlock the box. Each time they enter an incorrect code, the strength of the box decreases by 1. If the strength of the box becomes 0, the box automatically unlocks.

The class Box has a method `Unlock()` that:

- takes the code entered as a parameter and checks if it matches the code to unlock the box
 - returns `TRUE` if the parameter matches the unlock code
 - subtracts 1 from `Strength` if the parameter does not match the unlock code
 - checks if the new value of `Strength` is less than 1
 - returns `TRUE` if the new value of `Strength` is less than 1, otherwise it returns `FALSE`.

Write **program code** for the method `Unlock()`.

Programming language

Program code

[5]

[5]

- (c) The text file, Progress.txt, stores data about the player's previous progress.

The procedure LoadGame () :

- opens the text file in read mode
 - takes the data from the file and stores the data in the variable GameData
 - raises an exception with an appropriate message output if it cannot find the file.

Write **program code** for the procedure LoadGame () .

Programming language

Program code

[6]

[6]

- 8** A game stores details about characters.

A declarative programming language is used to represent the following knowledge base:

```

01 hair(blonde).

02 hair(black).

03 hair(red).

04 face(glasses).

05 face(moustache).

06 face(beard).

07 person(ismail).

08 person(anisha).

09 person(kim).

10 person(kyle).

11 has(kyle, glasses).

12 has(kyle, beard).

13 has(anisha, red).

14 has(kyle, black).

```

These clauses have the following meaning:

Clause	Explanation
01	Hair can be blonde
04	Glasses can be on the face
08	Anisha is a person
12	Kyle has a beard
13	Anisha has red hair

A person, x , is a selected person if they have black hair and either a moustache or a beard.

Write a rule to represent this condition.

SelectedPerson (X)

IF

.....
.....
.....
.....

[2]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

October/November 2015

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

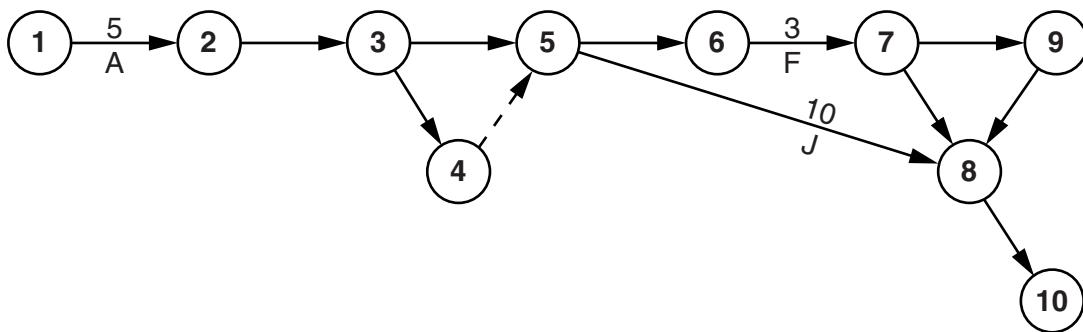
Complete the statement to indicate which high-level programming language you will use.

Programming language

- 1 A large software house has been asked to supply a computerised solution for a business. The project manager has drawn up a list of activities and their likely duration.

Activity	Description	Weeks to complete
A	Write requirement specification	5
B	Produce program design	5
C	Write module code	15
D	Module testing	10
E	Integration testing	5
F	Alpha testing	3
G	Install software and acceptance testing	5
H	Write end user training guide	5
J	Write technical documentation	10
K	End user training	4
L	Sign off final system	1

- (a) The project manager decides to construct a Program Evaluation Review Technique (PERT) chart from this data.



- (i) Complete the PERT chart.

[7]

- (ii) State the critical path.

..... [2]

- (iii) Calculate the minimum number of weeks for the completion of this solution.

..... [1]

(b) For activity J:

(i) State the earliest start time.

Week number [1]

(ii) State the latest start time.

Week number [1]

(c) Give a reason why the project manager used a PERT chart.

.....
..... [1]

- 2** A declarative programming language is used to represent the following facts and rules:

```

01 male(ali).
02 male(raul).
03 male(ahmed).
04 male(phiippe).
05 female(meena).
06 female(aisha).
07 female(gina).
08 parent(ali, raul).
09 parent(meena, raul).
10 parent(ali, ahmed).
11 parent(meena, ahmed).
12 parent(ali, aisha).
13 parent(meena, aisha).
14 father(A, B) IF male(A) AND parent(A, B).

```

These clauses have the following meaning:

Clause	Explanation
01	Ali is male
05	Meena is female
08	Ali is a parent of Raul
14	A is the father of B if A is male and A is a parent of B

- (a)** More facts are to be included.

Philippe and Gina are the parents of Meena.

Write the additional clauses to record this.

15

16, [2]

- (b)** Using the variable P, the goal

parent(P, raul)

returns

P = ali, meena

Write the result returned by the goal

parent(ali, C)

C =, [2]

- (c) Use the variable F to write the goal to find the father of Ahmed.

..... [1]

- (d) Write the rule to show that X is the mother of Y .

$\text{mother}(X, Y)$

IF

..... [2]

- (e) W is a grandparent of Z if W is a parent of one of Z 's parents.

Complete the following rule:

$\text{grandparent}(W, Z)$

IF

..... [2]

- (f) Complete the rule to show that G is a grandfather of K .

$\text{grandfather}(G, K)$

IF

..... [2]

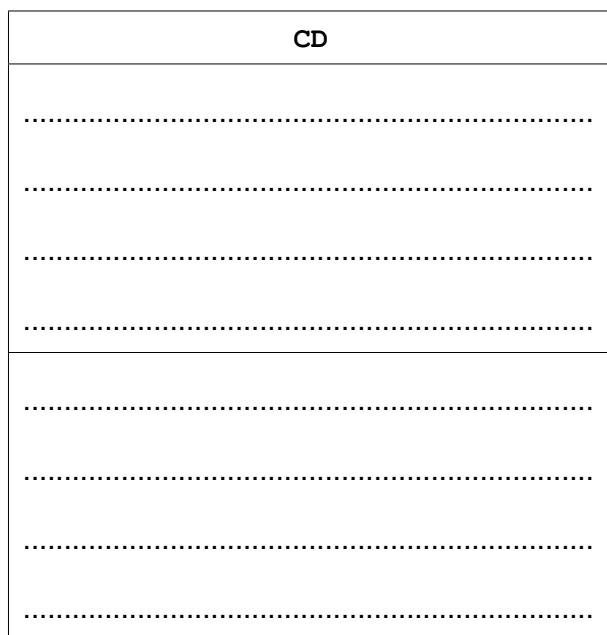
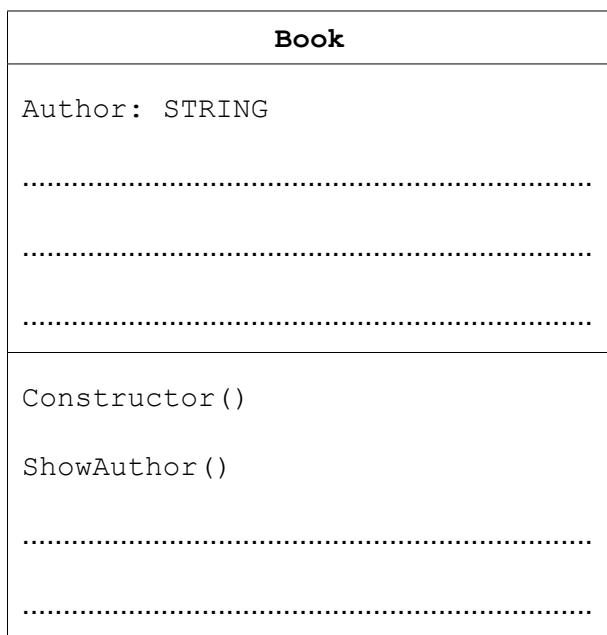
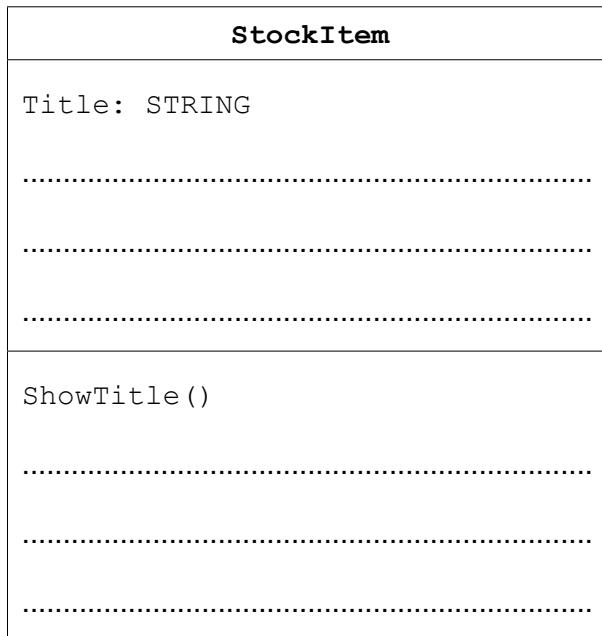
- 3 A lending library stocks two types of item for loan: books and CDs.

All stock items have a title, the date the item was acquired and whether the item is currently out on loan.

Books have an author and ISBN. CDs have an artist and play time in minutes.

The library needs a program to process data about the stock items. The program will use an object-oriented programming language.

- (a) Complete the class diagram showing the appropriate properties and methods.



(b) Write program code

- (i) for the class definition for the superclass StockItem.

- (ii) for the class definition for the subclass Book.

(iii) to create a new instance of Book with:

- identifier NewBook
- title "Computers"
- author A.Nyone
- ISBN 099111
- acquired on 12/11/2001
- not out on loan

Programming language
.....
.....
.....
.....
.....
.....
.....
..... [3]

Question 4 begins on page 10.

4 A binary tree Abstract Data Type (ADT) has these associated operations:

- create the tree (CreateTree)
- add an item to tree (Add)
- output items in ascending order (TraverseTree)

(a) Show the final state of the binary tree after the following operations are carried out.

```
CreateTree  
Add("Dodi")  
Add("Farai")  
Add("Elli")  
Add("George")  
Add("Ben")  
Add("Celine")  
Add("Ali")
```

[4]

- (b) The binary tree ADT is to be implemented as an array of nodes. Each node consists of data and two pointers.

Using pseudocode, a record type, Node, is declared as follows:

```
TYPE Node
    DECLARE Name : STRING
    DECLARE LeftPointer : INTEGER
    DECLARE RightPointer : INTEGER
ENDTYPE
```

The statement

```
DECLARE Tree : ARRAY[1:10] OF Node
```

reserves space for 10 nodes in array Tree.

The CreateTree operation links all nodes into a linked list of free nodes. It also initialises the RootPointer and FreePointer.

Show the contents of the Tree array and the values of the two pointers, RootPointer and FreePointer, after the operations given in part (a) have been carried out.

Tree			
RootPointer	Name	LeftPointer	RightPointer
[1]			
[2]			
[3]			
[4]			
[5]			
[6]			
[7]			
[8]			
[9]			
[10]			

[7]

- (c) A programmer needs an algorithm for outputting items in ascending order. To design this, the programmer writes a recursive procedure in pseudocode.

- (i) Complete the pseudocode:

```

01 PROCEDURE TraverseTree (BYVALUE Root: INTEGER)

02     IF Tree[Root].LeftPointer .....
```

```
03         THEN
```

```
04             TraverseTree (.....)
```

```
05         ENDIF
```

```
06         OUTPUT ..... Name
```

```
07         IF ..... <> 0
```

```
08             THEN
```

```
09                 TraverseTree (.....)
```

```
10             ENDIF
```

```
11 ENDPROCEDURE
```

[5]

- (ii) Explain what is meant by a recursive procedure. Give a line number from the code above that shows procedure `TraverseTree` is recursive.

.....
.....
.....

Line number [2]

- (iii) Write the pseudocode call required to output all names stored in `Tree`.

.....
.....

[1]

Question 5 begins on page 14.

- 5 Data about sports club members are stored in a random file of records.

- The key field of a member record is the member ID (range 1000 to 9999).
- Other member data are stored.
- A hashing function is used to calculate a record address.
- The random file initially consists of dummy records.
- Dummy records are shown by member ID set to 0.

```
FUNCTION Hash(MemberID : INTEGER) RETURNS INTEGER

    Address ← MemberID MOD 100

    RETURN Address

ENDFUNCTION
```

- (a) New members with the following member IDs have joined the sports club:

1001, 3005, 4096, 2098, 7002

Indicate where each record should be stored by deleting the zero and writing the member ID in the correct cell.

MembershipFile		
Address	MemberID	Other member data
0	0	
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
:		
:		
96	0	
97	0	
98	0	
99	0	

[2]

- (b) (i)** The program stores a new member's data in the record variable `NewMember`. The field `MemberID` stores the member ID.

Complete the pseudocode:

```

10 // generate record address
20 NewAddress ← .....
30 // move pointer to the disk address for the record
40 SEEK .....
50 PUTRECORD "MembershipFile", .....

```

[4]

- (ii)** Before records can be saved to the file `MembershipFile`, the file needs to be opened.

Complete the pseudocode.

```

01 TRY
02      OPENFILE ..... FOR RANDOM
03 EXCEPT
04      .....
05 ENDTRY

```

[2]

- (iii)** A record with member ID 9001 is to be stored.

Explain the problem that occurs when this record is saved.

.....

[2]

- (iv)** Describe a method, without changing the function `Hash`, to handle the problem identified in part (b)(iii).

.....

[2]

- (v) Write **pseudocode** to implement the method you described in part (b)(iv).

Choose line numbers to indicate where your pseudocode should be inserted in the pseudocode of **part (b)(i)**.

. [4]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/41

Paper 4 Practical

October/November 2022

2 hours 30 minutes

You will need: Candidate source files (listed on page 2)
evidence.doc



INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the evidence document, **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: **evidence_zz999_999**

A class declaration can be used to declare a record.

If the programming language does not support arrays, a list can be used instead.

A source file is used to answer **Question 1**. The file is called **IntegerData.txt**

- 1 The text file `IntegerData.txt` stores 100 integer numbers between 1 and 100 inclusive. A program is required to read in this data and perform searching and sorting on the data.

- (a) Write program code to declare a global 1D array, `DataArray`, with space for 100 integer values.

Save your program as **Question1_N22**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[2]

- (b) The procedure `ReadFile()` must read in the numbers from the text file and store each one in the array. Use appropriate exception handling.

Write program code for the procedure `ReadFile()`.

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[6]

- (c) The function `FindValues()` asks the user to enter a number to search for in the array. The number input must be a whole number between 1 and 100 inclusive. The function then returns the number of times the number input appears in the array.

Write program code for the function `FindValues()`.

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[7]

- (d) (i) Write program code to call `ReadFile()` and `FindValues()` from the main program. The return value from `FindValues()` must be output with an appropriate message.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[3]

- (ii) Test your program using the number 61 as input.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 1(d)(ii)** in the evidence document.

[1]

- (e) The procedure `BubbleSort()` needs to perform a bubble sort on the array and print the contents of the sorted array.

Write program code for the procedure `BubbleSort()` **and** call it from the main program.

Save your program.

Copy and paste the program code into **part 1(e)** in the evidence document.

[4]

- 2 A computer program is being developed that uses a set of cards. The program is written using object-oriented programming.

The program has two classes: Card and Hand.

The methods and attributes of these classes are shown:

Card	
Number : INTEGER	stores the card number from 1 to 5 inclusive
Colour : STRING	stores the card colour: red, blue or yellow
Constructor()	takes a number and colour as parameters and sets the private values to these parameters
GetNumber()	returns the card number
GetColour()	returns the card colour

Hand	
Cards : ARRAY[0:9] OF Card	1D array of type Card
FirstCard : INTEGER	stores the position of the first card in the hand
NumberCards : INTEGER	stores the number of cards in the hand
Constructor()	takes five card objects as parameters, assigns each card to the array Cards[], initialises FirstCard to 0 and NumberCards to 5
GetCard()	takes an index as a parameter and returns the card at that index in the array

- (a) (i) Write program code to declare the class Card, its attributes and constructor.

Do **not** write program code for the get methods.

Use your programming language appropriate constructor.

All attributes must be private. If you are writing in Python, include attribute declarations using comments.

Save your program as **Question2_N22**.

Copy and paste the program code into **part 2(a)(i)** in the evidence document.

[5]

- (ii) Write program code for the class methods GetNumber() and GetColour().

Save your program.

Copy and paste the program code into **part 2(a)(ii)** in the evidence document.

[3]

(iii) The program is tested with the following cards:

Number	Colour
1	red
2	red
3	red
4	red
5	red
1	blue
2	blue
3	blue
4	blue
5	blue
1	yellow
2	yellow
3	yellow
4	yellow
5	yellow

Write program code to declare each of these cards as a variable of type `Card` in the main program.

Save your program.

Copy and paste the program code into **part 2(a)(iii)** in the evidence document.

[2]

- (b) (i) Write program code to declare the class `Hand`, its attributes and constructor.

Do **not** write the get methods.

Use your programming language appropriate constructor.

All attributes must be private. If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into **part 2(b)(i)** in the evidence document.

[6]

- (ii) The get method `GetCard()` takes an index as a parameter and returns the card stored at that index in the array.

Write program code for the method `GetCard()`.

Save your program.

Copy and paste the program code into **part 2(b)(ii)** in the evidence document.

[2]

- (iii) Two players are declared with 5 cards each.

Player 1 has the cards: 1 red, 2 red, 3 red, 4 red, 1 yellow.

Player 2 has the cards: 2 yellow, 3 yellow, 4 yellow, 5 yellow, 1 blue.

Write program code to declare player 1 and player 2 as objects of type `Hand`, with the cards indicated.

Save your program.

Copy and paste the program code into **part 2(b)(iii)** in the evidence document.

[2]

(c) The function `CalculateValue()` takes a player's hand as a parameter and returns a score calculated using the following rules:

- If a card is red, 5 points are added to the player's score.
- If a card is blue, 10 points are added to the player's score.
- If a card is yellow, 15 points are added to the player's score.
- The number of each card in the hand is added to the player's score.

(i) Write program code for the function `CalculateValue()`.
Assume that there are only 5 cards in the player's hand in this function.

Save your program.

Copy and paste the program code into **part 2(c)(i)** in the evidence document.

[6]

(ii) Amend the main program by writing program code to use the function `CalculateValue()` for each of the two players. The player with the highest score wins.

Output an appropriate message to identify the winning player, or if the game was a draw (both players have the same number of points).

Save your program.

Copy and paste the program code into **part 2(c)(ii)** in the evidence document.

[4]

(iii) Test your program.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 2(c)(iii)** in the evidence document.

[1]

- 3 A binary tree consists of nodes. Each node has 3 integer values: a left pointer, data and a right pointer.

The binary tree is stored using a global 2D array.

The pseudocode declaration for the array is:

```
DECLARE ArrayNodes : ARRAY[0:19, 0:2] OF INTEGER
```

For example:

- `ArrayNodes[0, 0]` stores the left pointer for the first node.
- `ArrayNodes[0, 1]` stores the data for the first node.
- `ArrayNodes[0, 2]` stores the right pointer for the first node.

`-1` indicates a null pointer, or null data.

(a) Write program code to:

- declare the global 2D array `ArrayNodes`
- initialise all 3 integer values to `-1` for each node.

Save your program as **Question3_N22**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[3]

(b) The binary tree stores the following values:

Index	Left pointer	Data	Right pointer
0	1	20	5
1	2	15	-1
2	-1	3	3
3	-1	9	4
4	-1	10	-1
5	-1	58	-1
6	-1	-1	-1

`FreeNode` stores the index of the first free element in the array, initialised to 6.

`RootPointer` stores the index of the first node in the tree, initialised to 0.

Amend your program by writing program code to store the given data in `ArrayNodes` **and** initialise the free node and root node pointers.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[2]

- (c) The following recursive pseudocode function searches the binary tree for a given value. If the value is found, the function must return the index of the value. If the value is not found, the function must return -1.

The function is incomplete. There are **four** incomplete statements.

```

FUNCTION SearchValue(Root : INTEGER,
                     ValueToFind : INTEGER) RETURNS INTEGER

    IF Root = -1 THEN
        RETURN -1

    ELSE
        IF ArrayNodes[Root, 1] = ValueToFind THEN
            RETURN .....
        ELSE
            IF ArrayNodes[Root, 1] = -1 THEN
                RETURN -1
            ENDIF
        ENDIF
        ENDIF
        IF ArrayNodes[Root, 1] ..... ValueToFind THEN
            RETURN SearchValue(ArrayNodes[....., 0], ValueToFind)
        ENDIF
        IF ArrayNodes[Root, .....] < ValueToFind THEN
            RETURN SearchValue(ArrayNodes[Root, 2], ValueToFind)
        ENDIF
    ENDFUNCTION

```

Write program code for the function `SearchValue()`.

Save your program.

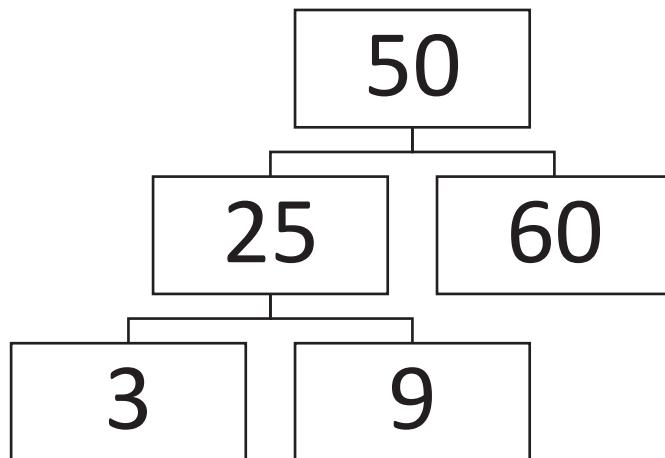
Copy and paste the program code into **part 3(c)** in the evidence document.

[5]

(d) A post order traversal performs the following operation:

- visit the left node
- visit the right node
- output the root.

For example, in the following tree, the output would be: 3 9 25 60 50



An outline of the `PostOrder()` procedure is:

- If left node is not empty, make a recursive call with the left node as the root.
- If right node is not empty, make a recursive call with the right node as the root.
- Output the current root node.

The procedure `PostOrder()` takes the root node as a parameter.

Write program code for the procedure `PostOrder()`.

Save your program.

Copy and paste the program code into **part 3(d)** in the evidence document.

[7]

(e) (i) Amend the main program by writing program code to:

- call the function `SearchValue()` to find the position of the number 15 in the tree
- use the result from `SearchValue()` to output either the index of the value if found, or an appropriate message to state that the value was not found
- call the procedure `PostOrder()`.

Save your program.

Copy and paste the program code into **part 3(e)(i)** in the evidence document.

[3]

(ii) Test your program.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 3(e)(ii)** in the evidence document.

[1]

BLANK PAGE

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.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

May/June 2018

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **17** printed pages and **3** blank pages.

- 1 A declarative language is used to represent facts and rules about flights.

```

01 direct(edinburgh, paris).
02 direct(palma, rome).
03 direct(glasgow, palma).
04 direct(glasgow, vienna).
05 direct(glasgow, salzburg).
06
07 flies(paris, fly_jet).
08 flies(mumbai, british_air).
09 flies(palma, ciebe).
10 flies(vienna, fly_jet).
11 flies(salzburg, ciebe).
12
13 can_fly(X, Y) IF direct(X, Z) AND direct(Z, Y).

```

These clauses have the following meaning:

Clause	Explanation
01	There is a direct route from Edinburgh to Paris.
07	Fly Jet operates flights to Paris.
13	It is possible to fly from X to Y if there is a direct flight from X to Z and a direct flight from Z to Y.

- (a) More facts need to be included.

There is a direct flight from London to Rome and British Air flies to Rome.

14

15

[2]

- (b) Using the variable Q, the goal

flies(Q, fly_jet).

returns

Q = paris, vienna

Write the result returned by the goal

flies(K, ciebe).

K = [2]

- (c) Use the variable M to write the goal to find where you can fly direct from Glasgow.

..... [2]

- (d) If an airline flies to an airport, that airline also flies every direct route out of that airport.

Write a rule to represent this condition.

`flies(Y, X)`

IF

..... [3]

- (e) State what the following goal returns.

`can_fly(glasgow, rome).`

..... [1]

- 2 The array `ItemList[1:20]` stores data. A **bubble sort** sorts these data.

- (a) Complete the pseudocode algorithm for a bubble sort.

```

01  MaxIndex ← 20
02  NumberItems ← .....
03  FOR Outer ← 1 TO .....
04    FOR Inner ← 1 to NumberItems
05      IF ItemList[Inner] > .....
06        THEN
07          Temp ← ItemList[.....]
08          ItemList[Inner] ← ItemList[.....]
09          ItemList[Inner + 1] ← .....
10        ENDIF
11    ENDFOR
12    NumberItems ← .....
13  ENDFOR

```

[7]

- (b) The algorithm in part (a) is inefficient.

- (i) Explain why the algorithm in part (a) is inefficient.

.....
.....
.....
.....

[2]

- (ii) Explain how you would improve the efficiency of this algorithm.

.....
.....
.....
.....

[3]

- (c) An insertion sort is another sorting algorithm.

State **two** situations when an insertion sort is more efficient than a bubble sort. Give a reason for each.

Situation 1

.....

Reason

.....

.....

Situation 2

.....

Reason

.....

.....

[4]

- 3 An internet based music streaming service provides access to an unlimited number of songs for members to play.

The following pseudocode represents the operation of the service.

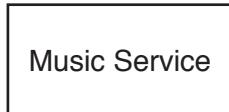
```

CALL OpenAccount()
CALL OperateAccount()
CALL CloseAccount()

PROCEDURE OperateAccount()
    WHILE RequestCloseAccount() = FALSE
        IF SubscriptionDue() = TRUE
            THEN
                CALL MakePayment()
            ELSE
                CALL PlaySong()
            ENDIF
        ENDWHILE
    ENDPROCEDURE

```

- (a) Complete the JSP structure diagram for this music service from the pseudocode given.



[5]

(b) The service needs extending so that members can download songs to play offline.

- When a member selects a song, the service checks if the song has already been downloaded.
- If the member has already downloaded the song, the member has the option to delete or play it.
- If the member has not already downloaded the song they have the option to download or stream it.

Complete the following JSP structure diagram to represent these new requirements.

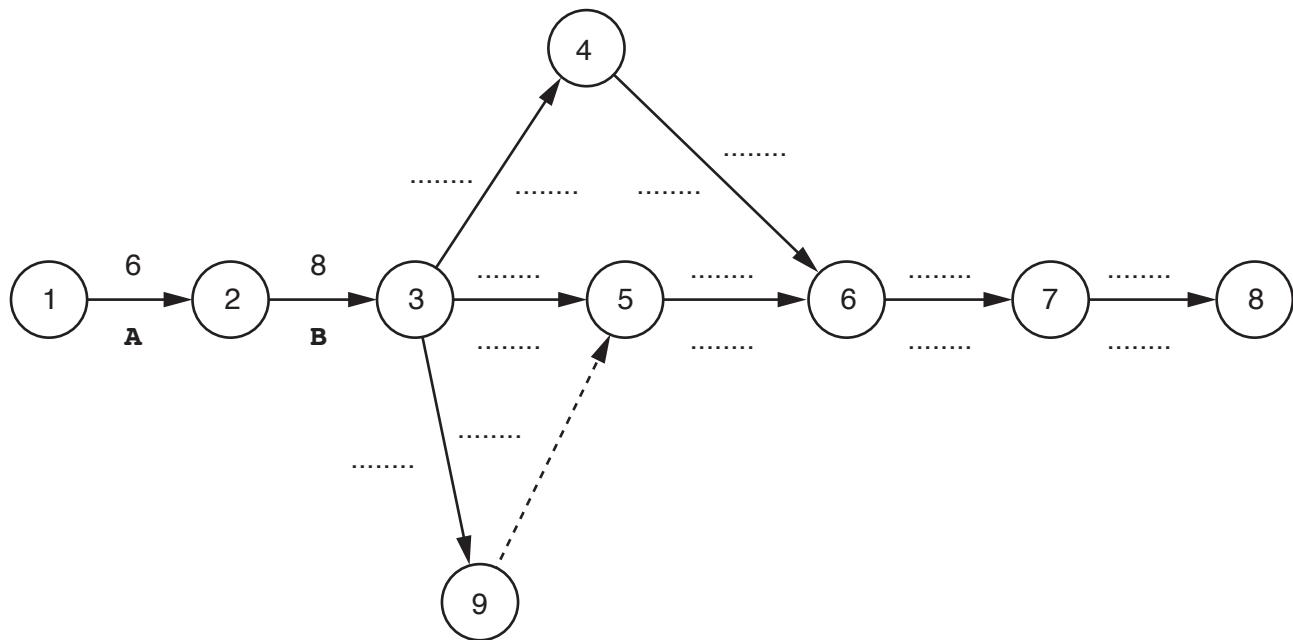


[4]

- 4 A software company is developing a new application. The project manager has created a work breakdown structure, as shown in the following table.

Activity		Days to complete	Predecessor
A	Gather user requirements	6	
B	Design work	8	A
C	Develop server code	4	B
D	Develop application code	5	B
E	User interface development	6	B
F	Test server code	2	C
G	Test application	2	D, E
H	Test application/server integration	5	F, G
I	Roll out mobile application	3	H

- (a) Use the data in the table to complete the following Program Evaluation Review Technique (PERT) chart.



[5]

- (b) Calculate the critical path (CP). State the:

activities that form the CP

duration of the CP

[2]

- (c) For activity F, state the:

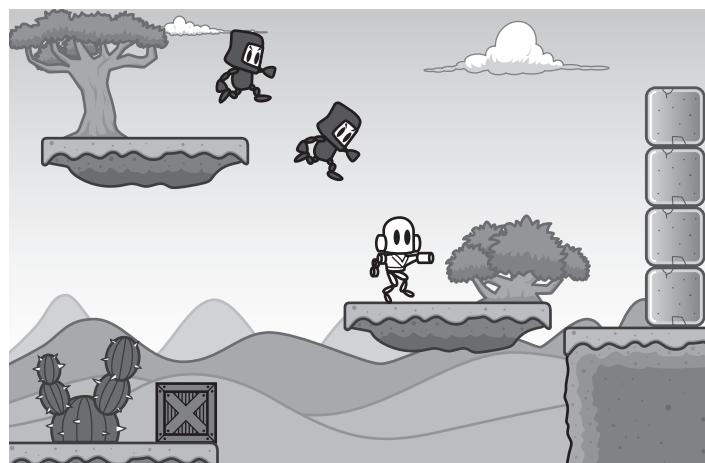
earliest start time

latest finish time

[2]

Question 5 begins on the next page.

- 5 A computer game is being developed using object-oriented programming. The following image is a screenshot from the game.



There are scenery elements and animated elements. The player's character is one of the animated elements.

Each game element has the attributes:

Attribute	Description	Example value
PositionX	The x coordinate of the game element.	92
PositionY	The y coordinate of the game element.	106
Width	The width of the game element.	150
Height	The height of the game element.	200
ImageFilename	The filename of the image file for the game element.	GameElementFrame1.png

Each game element has a method, `GetDetails()` that returns a string containing all the element's attributes.

The player's character is one of a number of animated elements. All animated elements have the attributes:

Attribute	Description	Example value
AnimationFrames	An array of GameElement	
Direction	A string giving the direction the object is travelling in.	"Left"
Strength	A value for the strength that indicates the power of the object.	2000
Health	A value for the health that indicates the health of the object.	100

The player's character can either move left or right, or jump.

- (a) Complete the following class diagram for the game.

You do not need to include any additional get or set methods.

GameElement
PositionX: INTEGER PositionY: INTEGER Width: INTEGER Height: INTEGER ImageFilename: STRING
Constructor() GetDetails()

AnimatedElement
AnimationFrames: ARRAY OF GameElement
.....
.....
.....
Constructor() AdjustHealth() AdjustStrength() DisplayAnimation()

Scenery
CauseDamage: BOOLEAN DamagePoints: INTEGER
Constructor() GiveDamagePoints()

Player
.....
.....
.....
.....
.....

[3]

- (b)** Write **program code** to define the GameElement class.

Programming language

Program code

- (c) The Scenery() class has two attributes, CauseDamage and DamagePoints.

If the attribute CauseDamage is TRUE, then the scenery element can cause damage.

The method `GiveDamagePoints()` checks whether the object can cause damage. If the object can cause damage, the method returns the integer value of the `DamagePoints` attribute.

Write program code for the Scenery class.

Programming language

Program code

.. [6]

(d) A new scenery object, `GiftBox`, is to be created.

(i) The attributes of `GiftBox` are as follows:

Attribute	Value
PositionX	150
PositionY	150
Width	50
Height	75
ImageFilename	"box.png"
CauseDamage	TRUE
DamagePoints	50

Write **program code** to create an instance of `GiftBox`.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

[3]

- (ii) An additional method, GetScenery(), returns all the attributes of the Scenery class.

Write program code for the GetScenery() method.

You should use the `GetDetails()` method that the `Scenery` class inherits from the `GameElement` class.

Programming language

Program code

[3]

... [3]

- 6 An Abstract Data Type (ADT) is used to create a linked list. The linked list is created as an array of records. The records are of type `ListNode`.

An example of a record of `ListNode` is shown in the following table.

Data Field	Value
Player	"Alvaro"
Pointer	1

- (a) (i) Use **pseudocode** to write a definition for the record type, `ListNode`.

.....

[3]

- (ii) An array, `Scorers`, will hold 10 nodes of type `ListNode`. Use **pseudocode** to write an array declaration for this array. The lower bound subscript is 0.

..... [2]

- (b) The linked list stores `ListNode` records in alphabetical order of player. The last node in the linked list always has a `Pointer` value of -1. The position of the first node in the linked list is held in the variable `ListHead`.

After some processing, the array and variables are in the state as follows:

Scorers		
ListHead	Player	Pointer
0	"Alvaro"	1
1	"Antoine"	3
2	"Dimitri"	7
3	"Cristiano"	2
4	"Gareth"	5
5	"Graziano"	6
6	"Olivier"	8
7	"Erik"	4
8	"Yaya"	9
9	"Zoto"	-1

A **recursive** function traverses the linked list to search for a player.

An example of calling the function, using pseudocode, is:

```
Position ← SearchList("Gareth", ListHead)
```

Complete the following **pseudocode** to implement the function `SearchList()`.

The function will return a value of 99 when a player is not found.

```
FUNCTION SearchList(Find : STRING, Position : INTEGER) RETURNS INTEGER
    IF Scorer[Position].Player = .....
        THEN
            RETURN .....
    ELSE
        IF Scorer[Position].Pointer <> -1
            THEN
                Position ← SearchList(Find, ....)
                RETURN .....
        ELSE
            RETURN .....
        ENDIF
    ENDIF
ENDFUNCTION
```

[5]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

October/November 2016

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

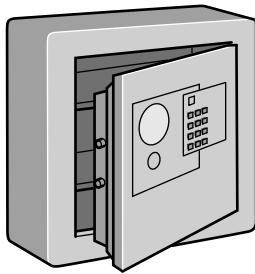
At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

- 1 A user can lock a safety deposit box by inputting a 4-digit code. The user can unlock the box with the same 4-digit code.



There is a keypad on the door of the safety deposit box. The following diagram shows the keys on the keypad.

1	2	3
4	5	6
7	8	9
R	0	Enter

Initially, the safety deposit box door is open and the user has not set a code.

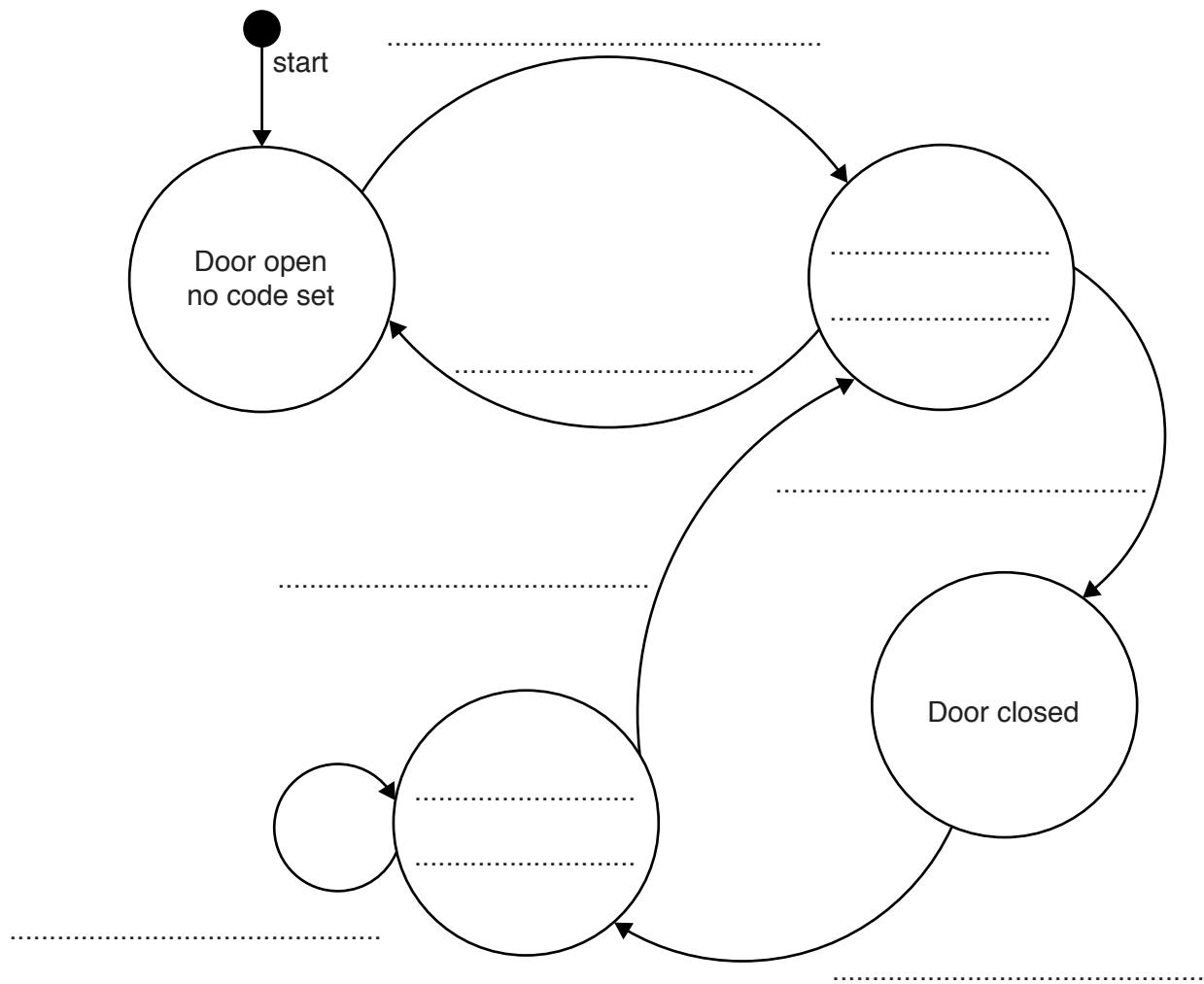
The operation of the safety deposit box is as follows:

- A) To set a new code the door must be open. The user chooses a 4-digit code and sets it by pressing the numerical keys on the keypad, followed by the Enter key. Until the user clears this code, it remains the same. (See point E below)
- B) The user can only close the door if the user has set a code.
- C) To lock the door, the user closes the door, enters the set code and presses the Enter key.
- D) To unlock the door, the user enters the set code. The door then opens automatically.
- E) The user clears the code by opening the door and pressing the R key, followed by the Enter key. The user can then set a new code. (See point A above)

The following state transition table shows the transition from one state to another of the safety deposit box:

Current state	Event	Next state
Door open, no code set	4-digit code entered	Door open, code set
Door open, code set	R entered	Door open, no code set
Door open, code set	Close door	Door closed
Door closed	Set code entered	Door locked
Door locked	Set code entered	Door open, code set
Door locked	R entered	Door locked

(a) Complete the state-transition diagram.



[7]

- (b) A company wants to simulate the use of a safety deposit box. It will do this with object-oriented programming (OOP).

The following diagram shows the design for the class `SafetyDepositBox`. This includes the properties and methods.

SafetyDepositBox	
Code	: STRING // 4 digits
State	: STRING // "Open-NoCode", "Open-CodeSet", "Closed" // or "Locked"
Create()	// method to create and initialise an object // if using Python use <code>__init__</code>
Reset()	// clears Code
SetState()	// set state to parameter value // and output new state
SetNewCode()	// sets Code to parameter value // output message and new code
StateChange()	// reads keypad and takes appropriate action

Write **program code** for the following methods.

Programming language

(i) Create()

.....
.....
.....
.....
.....

[3]

(ii) Reset()

.....
.....
.....

[2]

(iii) SetState()

[2]

[2]

(iv) SetNewCode()

[2]

[2]

(v) The user must enter a 4-digit code.

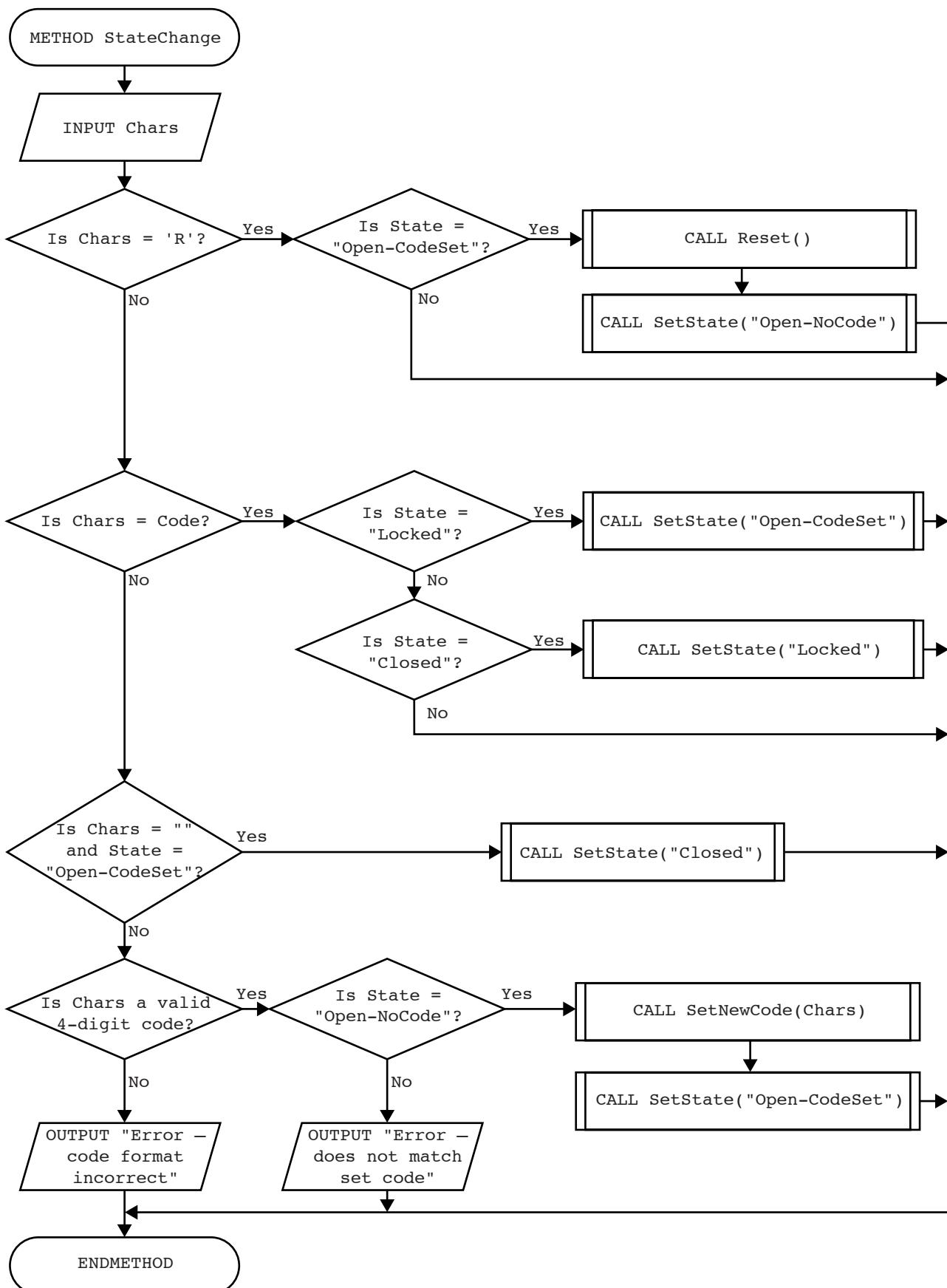
Write program code for a function Valid(s : STRING) that returns:

- TRUE if the input string s consists of exactly 4 digits
 - FALSE otherwise

Programming language

[4]

- (vi) Convert the flowchart to **program code** for the method `StateChange()`. Use the properties and methods in the original class definition and the `Valid()` function from part (v).



Programming language

[12]

- (vii) The company needs to write a program to simulate a safety deposit box. The program will create an object with identifier ThisSafe, which is an instance of the class SafetyDepositBox.

The main program design is:

```
instantiate ThisSafe (create and initialise ThisSafe)
loop forever (continually use ThisSafe)
    call StateChange() method
end loop
```

Write **program code** for the main program.

Programming language

[4]

- (c) It is possible to declare properties and methods as either public or private.

The programmer has modified the class design for `SafetyDepositBox` as follows:

SafetyDepositBox	
PRIVATE	
Code	: STRING
State	: STRING
PUBLIC	
Create()	
StateChange()	
PRIVATE	
Reset()	
SetState()	
SetNewCode()	

- (i) Describe the effects of declaring the `SafetyDepositBox` properties as private.

.....

[2]

- (ii) Describe the effects of declaring two methods of the class as public and the other three as private.

.....

[2]

2 Circle the programming language that you have studied:

Visual Basic (console mode) Python Pascal Delphi (console mode)

(a) (i) Name the programming environment you have used when typing in program code.

.....
.....

List **three** features of the editor that helped you to write program code.

1

.....

2

.....

3

..... [3]

(ii) Explain when and how your programming environment reports a syntax error.

When

.....

How

.....

[2]

Question 2 continues on page 12.

(iii) The table shows a module definition for BinarySearch in three programming languages.

Study one of the examples. Indicate your choice by circling A, B or C:

A **B** **C**

	A) Python
01	def BinarySearch(List, Low, High, SearchItem): Index = -1 while (Index == -1) AND (Low <= High): Middle = (High + Low) // 2 if List[Middle] == SearchItem: Index = Middle elif List[Middle] < SearchItem: Low = Middle + 1 else: High = Middle - 1 return(Middle)
	B) Pascal/Delphi
01	FUNCTION BinarySearch(VAR List : ARRAY OF INTEGER; Low, High, SearchItem : INTEGER) : INTEGER; 02 VAR Index, Middle : INTEGER; 03 BEGIN 04 Index := -1; 05 WHILE (Index = -1) & (Low <= High) DO 06 BEGIN 07 Middle := (High + Low) DIV 2; 08 IF List[Middle] = SearchItem 09 THEN Index := Middle 10 ELSE IF List[Middle] < SearchItem 11 THEN Low := Middle + 1 12 ELSE High := Middle - 1; 13 END; 14 Result := Middle; 15 END;
	C) Visual Basic
01	Function BinarySearch(ByRef List() As Integer, ByVal Low As Integer, ByVal High As Integer, ByVal SearchItem As Integer) As Integer 02 Dim Index, Middle As Integer 03 Index = -1 04 Do While (Index = -1) & (Low <= High) 05 Middle = (High + Low) \ 2 06 If List(Middle) = SearchItem Then 07 Index = Middle 08 ElseIf List(Middle) < SearchItem Then 09 Low = Middle + 1 10 Else 11 High = Middle - 1 12 End If 13 Loop 14 BinarySearch = Middle 15 End Function

The programming environment reported a syntax error in the `BinarySearch` code.

State the line number:

Write the correct code for this line.

..... [2]

- (b) (i) State whether programs written in your programming language are compiled or interpreted.

.....

[1]

- (ii) A programmer corrects the syntax error and tests the function. It does not perform as expected when the search item is not in the list.

State the type of error:

Write down the line number where the error occurs.

.....

Write the correct code for this line.

..... [2]

- (iii) State the programming environment you have used when debugging program code.

.....

Name **two** debugging features and describe how they are used.

1

.....

.....

2

.....

.....

[4]

- 3 The following table shows part of the instruction set for a processor which has one general purpose register, the Accumulator (ACC), and an index register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the given address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n into IX.
STO	<address>	Store the contents of ACC at the given address.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

A programmer is writing a program that outputs a string, first in its original order and then in reverse order.

The program will use locations starting at address NAME to store the characters in the string. The location with address MAX stores the number of characters that make up the string.

The programmer has started to write the program in the table opposite. The Comment column contains descriptions for the missing program instructions.

Complete the program using op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// initialise index register to zero
			// initialise COUNT to zero
LOOP1:			// load character from indexed address NAME
			// output character to screen
			// increment index register
			// increment COUNT starts here
			// is COUNT = MAX ?
			// if FALSE, jump to LOOP1
REVERSE:			// decrement index register
			// set ACC to zero
			// store in COUNT
LOOP2:			// load character from indexed address NAME
			// output character to screen
			// decrement index register
			// increment COUNT starts here
			// is COUNT = MAX ?
			// if FALSE, jump to LOOP2
			// end of program
COUNT:			
MAX:	4		
NAME:	B01000110		// ASCII code in binary for 'F'
	B01010010		// ASCII code in binary for 'R'
	B01000101		// ASCII code in binary for 'E'
	B01000100		// ASCII code in binary for 'D'

[15]

- 4** Commercial software usually undergoes acceptance testing and integration testing.

Distinguish between the two types of testing by stating:

- who does the testing
- when the testing occurs
- the specific purpose of each type of testing

(i) Acceptance testing

Who

.....

When

.....

Purpose

..... [3]

(ii) Integration testing

Who

.....

When

.....

Purpose

..... [3]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

CANDIDATE
NAME

--	--	--	--	--

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

October/November 2021

2 hours

You must answer on the question paper.

No additional materials are needed.

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.

This document has **20** pages. Any blank pages are indicated.

- 1 Sandy is writing a program to process data in a stack. The stack is implemented as a 1D array, DataStack, which has up to 100 elements.

The function Push (Value) stores Value on the stack and returns TRUE if Value was added to the stack, or FALSE if the stack is full.

The function Pop () returns the item at the top of the stack, or returns -1 if the stack is empty.

DataStack and TopPointer are declared as global.

- (a) Show the state of DataStack and its pointer after the following functions are executed on the current contents.

Pop ()

Pop ()

Push (19)

Pop ()

Push (50)

TopPointer	3	Index	Data
		[7]	
		[6]	
		[5]	
		[4]	
		[3]	8
		[2]	6
		[1]	20
		[0]	10
			[2]

- (b) Write **program code** for the function `Pop()`.

Programming language

Program code

[5]

(c) Sandy has also used a queue in her program.

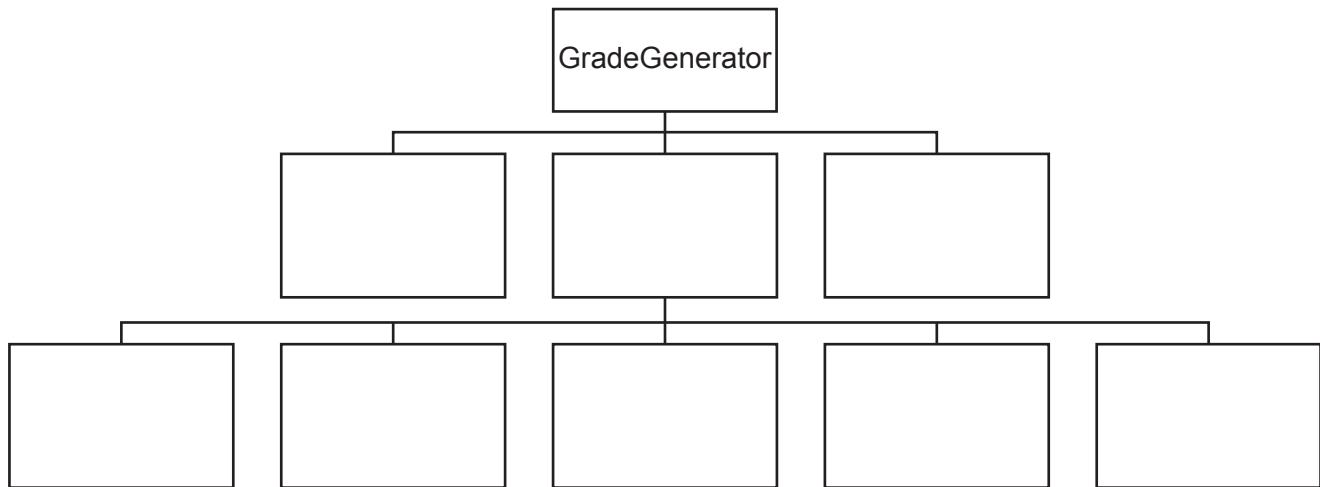
Describe the ways in which a queue differs from a stack.

.....
.....
.....
.....

- 2 A grade generator program takes the mark a student obtained in a test as input.

The program calculates and outputs the grade that matches the mark. The grade is either A, B, C, D or U.

Complete the following JSP structure diagram for the grade generator program.



[4]

- 3 The following pseudocode algorithm performs a binary search on the sorted array ThisArray.

The algorithm returns either the location of SearchItem in the array, or -1 if SearchItem is not in the array.

The function DIV returns the integer value of the division, for example, 11 DIV 2 returns 5.

Complete the algorithm by writing the missing pseudocode statements.

```

FUNCTION BinarySearch(ThisArray[], LowerBound, UpperBound,
                      SearchItem : INTEGER) RETURNS INTEGER

DECLARE Flag : BOOLEAN
DECLARE Mid : INTEGER
Flag ← -2
WHILE Flag <> -1
    Mid ← LowerBound + ((UpperBound - LowerBound) DIV 2)
    IF ..... < .....
        THEN
            RETURN .....
        ELSE
            IF ThisArray[Mid] > SearchItem
                THEN
                    UpperBound ← Mid .....
                ELSE
                    IF ThisArray[Mid] < SearchItem
                        THEN
                            LowerBound ← Mid .....
                        ELSE
                            RETURN .....
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDWHILE
ENDFUNCTION

```

[6]

- 4 Teachers in a school may work on Mondays, Tuesdays and Wednesdays. There are three time slots on each day: time slot 1, time slot 2 and time slot 3.

A teacher is either busy or free.

The school is using a declarative language to write a program to record which teachers are busy in each time slot on each day.

The following knowledge base is used:

```

01 teacher(james).
02 teacher(jill).
03 teacher(karl).
04 teacher(kira).
05 day(monday).
06 day(tuesday).
07 day(wednesday).
08 timeSlot(1).
09 timeSlot(2).
10 timeSlot(3).
11 busy(james, monday, 1).
12 busy(james, tuesday, 2).
13 busy(karl, monday, 1).
14 busy(kira, wednesday, 3).
```

These clauses have the following meaning:

Clause	Explanation
01	James is a teacher
05	Monday is a day
08	1 is a time slot
11	James is busy in time slot 1 on Monday

- (a) More facts need to be included.

Fred is a teacher who is busy in time slot 1 on Tuesday.

Write additional clauses for these facts.

15

16

[2]

- (b) Additional clauses are needed to identify whether Jill is busy in time slot 1 on Monday, Tuesday, or Wednesday.

Write these additional clauses.

17

18

19

[2]

- (c) Write a goal, using the variable x , to find all the teachers who are busy in time slot 3 on Monday.

.....

..... [1]

- (d) Write a rule to find whether a teacher x is free in a specific time slot y on day z .

IsTeacherFree(X, Z, Y)

IF

.....

.....

..... [4]

- 5 The recursive algorithm for the `Recursion()` function is defined in pseudocode as follows:

```

FUNCTION Recursion(A, B : INTEGER) RETURNS INTEGER

IF A <= 100

THEN

RETURN 1

ELSE

IF A > B

THEN

RETURN 5 + Recursion(A - 1, B)

ELSE

RETURN 10 + Recursion(A - 10, B)

ENDIF

ENDIF

ENDFUNCTION

```

- (a) The function is called with the following pseudocode statement:

```
OUTPUT Recursion(104, 102)
```

Dry run the function and complete the trace table. Give the output the program will produce.

Trace table:

Function call	A	B	Return value

Output =

Working

.....

.....

[4]

- (b) Rewrite the function `Recursion()` in **pseudocode**, using an **iterative algorithm**.

[4]

- 6 Kobi is writing an application that uses a record structure to store data.

- (a) (i) Describe what is meant by a **record structure**.

.....

 [2]

- (ii) The record structure stores the unique ID number (a whole number), first name and last name of a customer.

Write a **pseudocode** declaration for the record structure CustomerData.

.....

 [2]

- (b) Kobi's application stores the records in a random access file.

The function `StoreRecord()`:

- takes a customer record as a parameter
- uses the function `CustomerHash()` to calculate and return the hash value for its parameter
- stores the customer record in the returned hash value address.

Assume there are no collisions.

Complete the following pseudocode algorithm to write a new record to the random access file.

```
PROCEDURE StoreRecord(NewData : .....)

  HashValue ← CustomerHash(NewData.CustomerID)

  Filename ← "CustomerRecords.dat"

  OPENFILE Filename FOR .....

  SEEK Filename, .....

  PUTRECORD Filename, .....

  ..... Filename

ENDPROCEDURE
```

[5]

- (c) Identify **two** typical features of a debugger **and** describe how Kobi could use each one during the development of the application.

Feature 1

.....
.....
.....

Feature 2

.....
.....
.....

[4]

- (d) Give **one** benefit and **one** drawback of Kobi using a program generator whilst developing his application.

Benefit

.....

Drawback

[2]

- 7 Sonya is writing a computer program that requires a user input. The user should input an integer between 1 and 100. Sonya wants to use exception handling.

- (a) Explain the reasons why Sonya should use exception handling in her program.

.....
.....
.....
.....

[2]

- (b) Write **program code** to read in the number from the user and raise an exception if the data is not valid.

Programming language

Program code

.....
.....
.....
.....
.....
.....
.....
.....

[3]

- (c) Give **two other** examples of where exception handling can be used in a program.

1

2

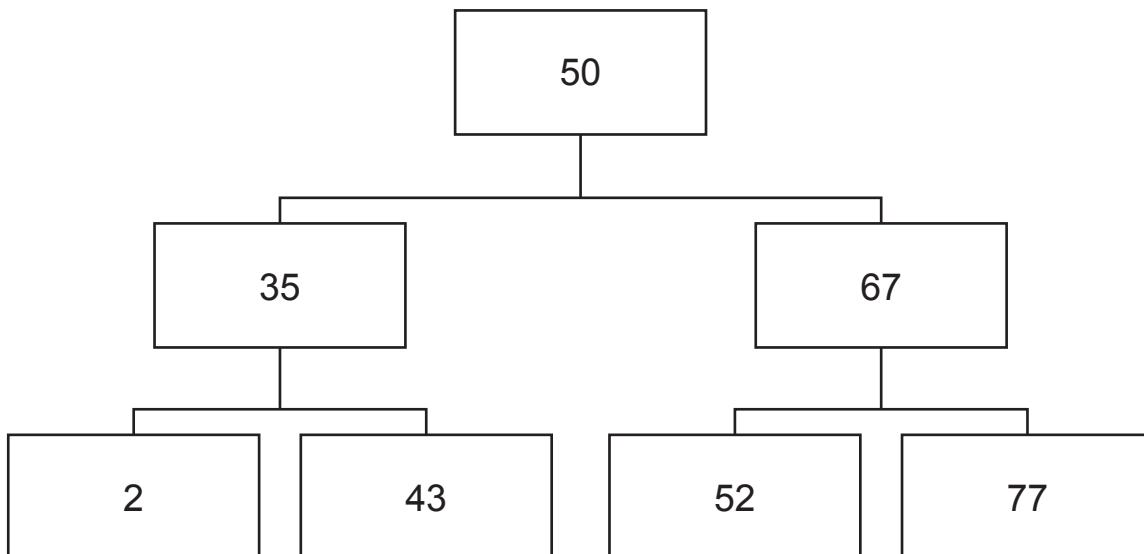
[2]

- 8 Data entered into a computer is stored in an ordered binary tree.

The binary tree is stored in a 2D array, `BinaryTree`.

The first element of the array is index 0.

- (a) The current contents of the binary tree are:



Complete the `LeftPointer` and `RightPointer` values in the following table for the binary tree shown.

A null pointer is represented by -1.

RootNode	0	Index	LeftPointer	Data	RightPointer
[0]				50	
[1]				67	
[2]				77	
[3]				35	
[4]				2	
[5]				43	
[6]				52	
[7]					
[8]					
[9]					
[10]					

[2]

- (b) A post-order tree traversal outputs the left node, then the right node, then the root node.

In the tree given in **part (a)**, the post-order tree traversal would output:

2 43 35 52 77 67 50

Complete the following recursive pseudocode algorithm `PostOrder()`.

```
PROCEDURE PostOrder (..... : INTEGER)

IF BinaryTree[RootNode, 0] <> -1

THEN
..... (BinaryTree[RootNode, .....])

ENDIF

IF BinaryTree[RootNode, 2] <> -1

THEN
..... (BinaryTree[RootNode, 2])

ENDIF

OUTPUT BinaryTree[RootNode, .....]

ENDPROCEDURE
```

[5]

- 9 A program uses a hashing algorithm to store data in the global array, `StoredData`.

The first element of the array is index 0. The array has 10000 integer elements.

- (a) Write a **pseudocode** declaration for the array `StoredData` and initialise each element to -1.

[3]

[3]

- (b) The hashing algorithm calculates the remainder after dividing the data by 1000, and then adds 6 to it.

The function `AddItem()` takes the data as a parameter. It calculates the index to store the data using the hashing algorithm.

If there is a collision, the function:

- checks the next index until it finds an index that does not have data in it
 - continues to search from the start of the array, if it reaches the end of the array.

The function returns `TRUE` if the item was successfully added, and `FALSE` if the array is full.

Write **program code** for the function AddItem().

Programming language

Program code

[7]

[7]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

October/November 2019

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

- 1 Each student at CIE University needs a printing account to print documents from university computers.

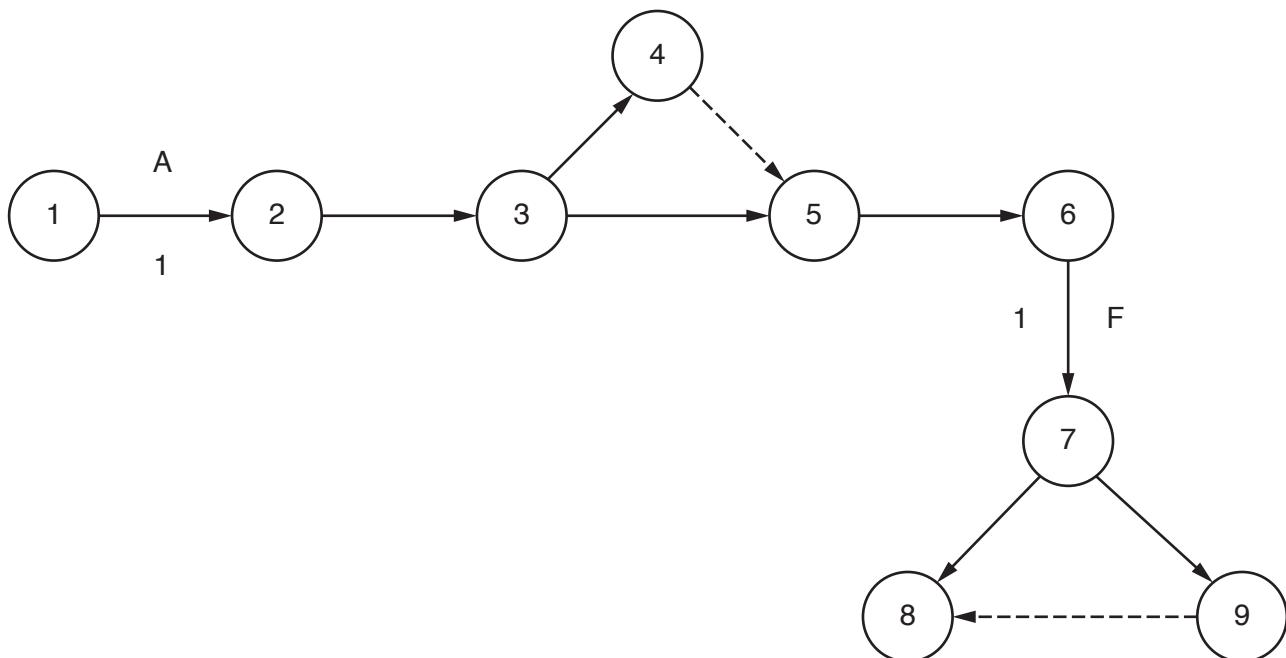
The university is developing software to manage each student's printing account and the printing process.

- (a) Developing the software will include the following activities.

Activity	Description	Time in weeks	Predecessor
A	Identify requirements	1	-
B	Produce design	3	A
C	Write code	10	B
D	Test modules	7	B
E	Final system black-box testing	3	C, D
F	Install software	1	E
G	Acceptance testing	2	F
H	Create user documentation	2	F

- (i) Add the correct activities and times to the following Program Evaluation Review Technique (PERT) chart for the software development.

Two activities and times have been done for you.



[6]

- (ii) State what is meant by the **critical path** in a PERT chart.

.....
.....

[1]

- (iii) Identify **and** describe a project planning technique, other than a PERT chart.

.....
.....
.....

[2]

- (b) When a student prints a document, a print job is created. The print job is sent to a print server.

The print server uses a queue to hold each print job waiting to be printed.

- (i) The queue is circular and has six spaces to hold jobs.

The queue currently holds four jobs waiting to be printed. The jobs have arrived in the order A, B, D, C.

Complete the diagram to show the current contents of the queue.



[1]

- (ii) Print jobs A and B are now complete. Four more print jobs have arrived in the order E, F, G, H.

Complete the diagram to show the current contents and pointers for the queue.



[3]

- (iii) State what would happen if another print job is added to the queue in the status in part (b)(ii).

.....
.....

[1]

- (iv) The queue is stored as an array, Queue, with six elements. The following algorithm removes a print job from the queue and returns it.

Complete the following **pseudocode** for the function Remove.

```

FUNCTION Remove RETURNS STRING
    DECLARE PrintJob : STRING
    IF ..... = EndPointer
        THEN
            RETURN "Empty"
        ELSE
            PrintJob ← Queue[.....]
            IF StartPointer = .....
                THEN
                    StartPointer ← .....
                ELSE
                    StartPointer ← StartPointer + 1
                ENDIF
            RETURN PrintJob
        ENDIF
    ENDFUNCTION

```

[4]

- (v) Explain why the circular queue could not be implemented as a stack.

.....

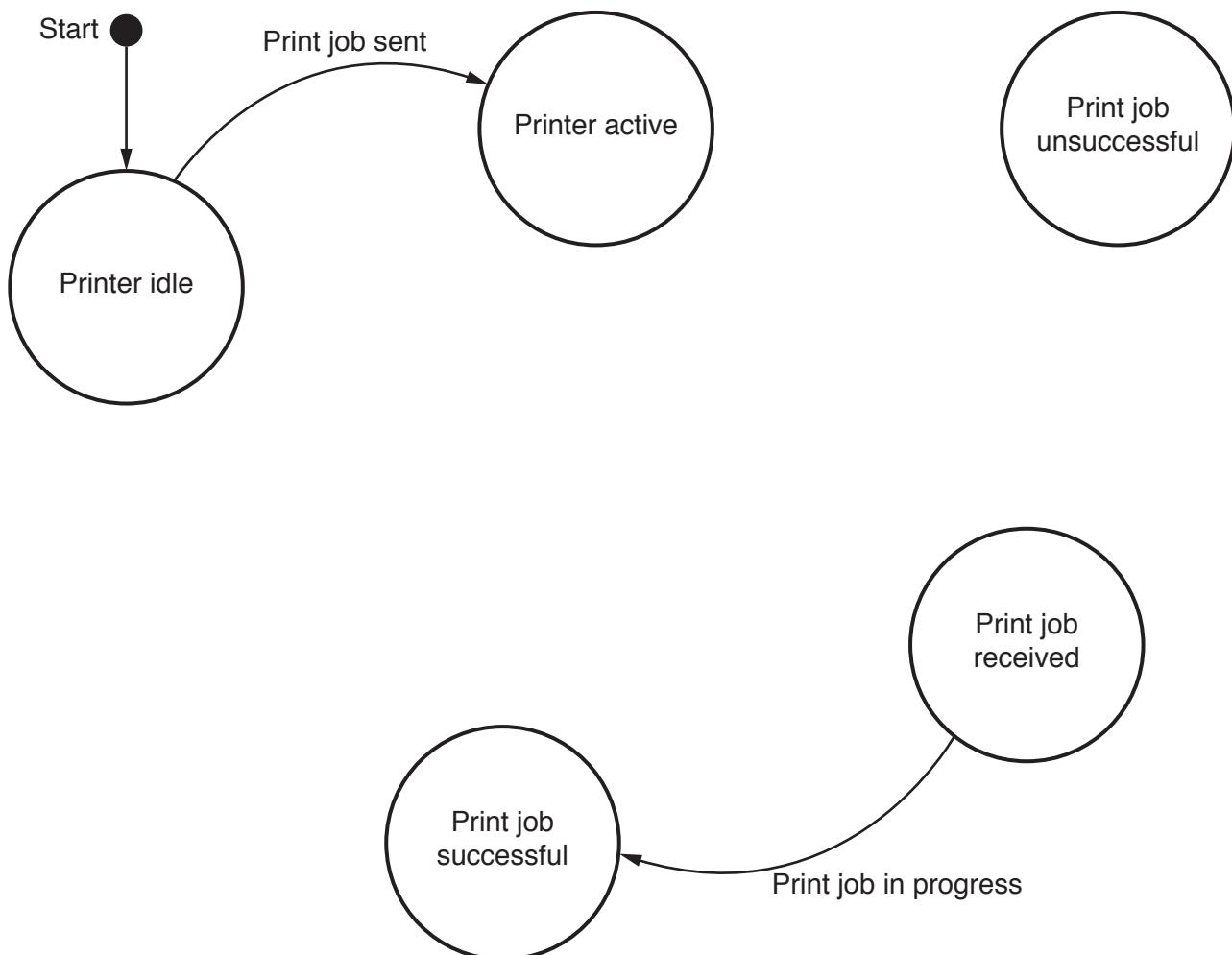
[2]

- (c) The university wants to analyse how a printer and a print server deal with the print jobs.

The following table shows the transitions from one state to another for the process.

Current state	Event	Next state
Printer idle	Print job sent	Printer active
Printer active	Print job added to queue	Print job received
Print job received	Print job in progress	Print job successful
Print job received	Print job in progress	Print job unsuccessful
Print job successful	Check print queue	Printer active
Print job unsuccessful	Error message displayed	Printer active
Printer active	Timeout	Printer idle

Complete the state-transition diagram for the table.



[5]

- (d) The university wants to assess troubleshooting issues with a printer. It wants to use a decision table to do this.

The troubleshooting actions are:

- check the connection from computer to printer, if the error light is flashing **and** the document has not been printed
- check the ink status, if the quality is poor
- check whether there is a paper jam, if the error light is flashing **and** the document has not been printed
- check the paper size selected, if the paper size is incorrect.

- (i) Describe the purpose of a decision table.
-
.....
.....
.....

[2]

- (ii) Complete the rules for the actions in the following decision table.

		Rules							
Conditions	Document printed but the quality is poor	Y	Y	Y	Y	N	N	N	N
	Error light is flashing on printer	Y	Y	N	N	Y	Y	N	N
	Document printed but paper size is incorrect	Y	N	Y	N	Y	N	Y	N
Actions	Check connection from computer to printer								
	Check ink status								
	Check if there is a paper jam								
	Check the paper size selected								

[4]

- (iii) Simplify your solution by removing redundancies.

		Rules						
Conditions	Document printed but the quality is poor							
	Error light is flashing on printer							
	Document printed but paper size is incorrect							
Actions	Check connection from computer to printer							
	Check ink status							
	Check if there is a paper jam							
	Check the paper size selected							

[5]

- (e) There are 1000 students at the university. They will each require a printing account.

Students need to buy printing credits that will be added to their account. Each page printed uses one printing credit.

The university needs software to keep track of the number of printing credits each student has in their account. The university has decided to implement the software using object-oriented programming (OOP).

The following diagram shows the design for the class `PrintAccount`. This includes the attributes and methods.

PrintAccount	
FirstName	: STRING // parameter sent to Constructor()
LastName	: STRING // parameter sent to Constructor()
PrintID	: STRING // parameter sent to Constructor()
Credits	: INTEGER // initialised to 50
Constructor()	// instantiates an object of the PrintAccount class, // and assigns initial values to the attributes
GetName()	// returns FirstName and LastName concatenated // with a space between them
GetPrintID()	// returns PrintID
SetFirstName()	// sets the FirstName for a student
SetLastName()	// sets the LastName for a student
SetPrintID()	// sets the PrintID for a student
AddCredits()	// increases the number of credits for a student
RemoveCredits()	// removes credits from a student account

- (i) Write **program code** for the `Constructor()` method.

Programming language

Program code

[4]

- (ii) Write **program code** for the SetFirstName () method.

Programming language

Program code

12

[2]

- (iii) Write **program code** for the GetName () method.

Programming language

Program code

[2]

- (iv) The method `AddCredits()` calculates the number of printing credits a student buys and adds the printing credits to the student's account.

- Credits cost \$1 for 25 credits.
 - If a student buys \$20 or more of credits in a single payment, they receive an extra 50 credits.
 - If a student buys between \$10 and \$19 (inclusive) of credits in a single payment, they receive an extra 25 credits.

Payment from a student is stored in the variable MoneyInput. This is passed as a parameter.

Write program code for AddCredits(). Use constants for the values that do not change.

Programming language

Program code

[6]

- (v) A global array, `StudentAccounts`, stores 1000 instances of `PrintAccount`.

Write **pseudocode** to declare the array `StudentAccounts`.

[2]

(vi) The main program has a procedure, CreateID(), that:

- takes the first name and last name as parameters
 - creates `PrintID` that is a concatenation of:
 - the first three letters of the first name in lower case
 - the first three letters of the last name in lower case
 - the character '1'
for example, the name Bill Smith would produce "bilsml1"
 - checks if the `PrintID` created already exists in the global array `StudentAccounts`:
 - If `PrintID` does not exist, it creates an instance of `PrintAccount` in the next free index in `StudentAccounts`.
 - If `PrintID` does exist, the number is incremented until a unique ID is created, for example, "bilsml2". It then creates an instance of `PrintAccount` in the next free index in `StudentAccounts`.

The global variable `NumberStudents` stores the number of print accounts that have currently been created.

Write program code for the procedure `CreateID()`. Do not write the procedure header.

Programming language

Program code

[8]

[8]

- 2 The following table shows part of the instruction set for a processor, which has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>. <address> can be an absolute address or a symbolic address.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

A programmer writes a program that multiplies two numbers together and outputs the result. The numbers are stored as NUMONE and NUMTWO.

The programmer has started to write the program in the following table. The comment column contains explanations for some of the missing program instructions and data.

Complete the program using the given instruction set.

Label	Op code	Operand	Comment
LOOP:			// load the value from ANSWER
			// add the value from NUMONE
			// load the value from COUNT
			// increment the Accumulator
			// is NUMTWO = COUNT ?
			// if false, jump to LOOP
			// load the value from ANSWER
			// output ANSWER to the screen
			// end of program
NUMONE:	2		
NUMTWO:	4		
COUNT:	0		
ANSWER:	0		

[9]

- 3 Software may not perform as expected. One reason for this is that a syntax error exists in the code.

Identify **three other** reasons why software may not perform as expected.

- 1
-
- 2
-
- 3
-

[3]

- 4 The following table contains definitions related to testing terminology.

Complete the table with the correct testing term to match the definition.

Definition	Term
Software is tested by an in-house team of dedicated testers.
Software is tested by the customer before it is signed off.
Software is tested by a small selection of users before general release.

[3]

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

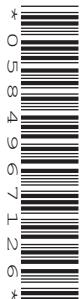
CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

May/June 2015

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language

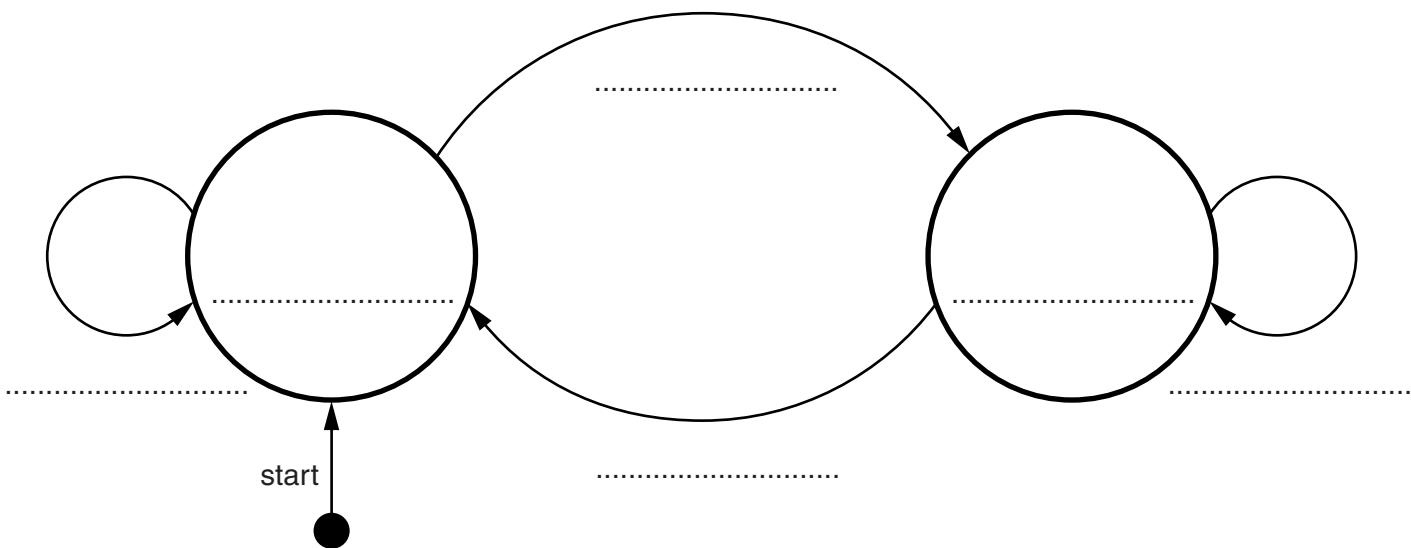
- 1 A turnstile is a gate which is in a locked state. To open it and pass through, a customer inserts a coin into a slot on the turnstile. The turnstile then unlocks and allows the customer to push the turnstile and pass through the gate.

After the customer has passed through, the turnstile locks again. If a customer pushes the turnstile while it is in the locked state, it will remain locked until another coin is inserted.

The turnstile has two possible states: **locked** and **unlocked**. The transition from one state to another is as shown in the table below.

Current state	Event	Next state
Locked	Insert coin	Unlocked
Locked	Push	Locked
Unlocked	Attempt to insert coin	Unlocked
Unlocked	Pass through	Locked

Complete the state transition diagram for the turnstile:



[5]

- 2** A declarative programming language is used to represent the knowledge base shown below:

```

01 capital_city(amman).
02 capital_city(beijing).
03 capital_city(brussels).
04 capital_city(cairo).
05 capital_city(london).
06 city_in_country(amman, jordan).
07 city_in_country(shanghai, china).
08 city_in_country(brussels, belgium).
09 city_in_country(london, uk).
10 city_in_country(manchester, uk).
11 country_in_continent(belgium, europe).
12 country_in_continent(china, asia).
13 country_in_continent(uk, europe).
14 city_visited(amman).
15 city_visited(beijing).
16 city_visited(cairo).

```

These clauses have the following meaning:

Clause	Explanation
01	Amman is a capital city
06	Amman is a city in the country of Jordan
11	Belgium is a country in the continent of Europe
14	The travel writer visited Amman

- (a)** More facts are to be included.

The travel writer visited the city of Santiago which is the capital city of Chile, in the continent of South America.

Write additional clauses to record this.

17

.....

18

.....

19

.....

20

.....

[4]

- (b) Using the variable ThisCountry, the goal

```
country_in_continent(ThisCountry, europe)
```

returns

```
ThisCountry = belgium, uk
```

Write the result returned by the goal:

```
city_in_country(ThisCity, uk)
```

ThisCity = [2]

- (c) Complete the rule below to list the countries the travel writer has visited.

```
countries_visited(ThisCountry)
```

IF [4]

.....

.....

.....

..... [4]

- 3 A shop gives some customers a discount on goods totalling more than \$20.
 The discounts are:
- 5% for goods totalling more than \$100
 - 5% with a discount card
 - 10% with a discount card and goods totalling more than \$100

(a) Complete the decision table.

Conditions	goods totalling more than \$20	Y	Y	Y	Y	N	N	N	N
	goods totalling more than \$100	Y	Y	N	N	Y	Y	N	N
	have discount card	Y	N	Y	N	Y	N	Y	N
Actions	No discount								
	5% discount								
	10% discount								

[4]

(b) Simplify your solution by removing redundancies.

Conditions	goods totalling more than \$20								
	goods totalling more than \$100								
	have discount card								
Actions	No discount								
	5% discount								
	10% discount								

[5]

- (c) The simplified table produced in part (b) is used as a design for program code.

The following identifier table shows the parameters to be passed to the function `Discount`. This function returns the discount amount as an integer.

Identifier	Data type
GoodsTotal	INTEGER
HasDiscountCard	BOOLEAN

Write **program code** for this function.

Programming language

[6]

- 4 A payroll program is to be written using an object-oriented programming language. An `Employee` class is designed. Two subclasses have been identified:

 - `HourlyPaidEmployee` who is paid a monthly wage calculated from their hourly rate of pay and the number of hours worked during the month
 - `SalariedEmployee` who is paid a monthly wage which is one 12th of their annual salary

(a) Draw an inheritance diagram for these classes.

[3]

- (b)** The design for the Employee class consists of:

 - properties
 - EmployeeName
 - EmployeeID
 - AmountPaidThisMonth
 - methods
 - SetEmployeeName
 - SetEmployeeID
 - CalculatePay

Write **program code** for the class definition of the superclass Employee.

Programming language

.....

.....

.....

.....

.....

.....

.....

.....

[5]

- (c) (i) State the properties and/or methods required for the subclass HourlyPaidEmployee.

.....
.....
.....
..... [4]

- (ii) State the properties and/or methods required for the subclass SalariedEmployee.

.....
.....
.....
..... [2]

- (d) Name the feature of object-oriented program design that allows the method CalculatePay to be declared in the superclass Employee.

.....
..... [1]

- 5 Data is stored in the array NameList[1:10]. This data is to be sorted.

- (a) (i) Complete the pseudocode algorithm for an insertion sort.

```

FOR ThisPointer ← 2 TO .....
    // use a temporary variable to store item which is to
    // be inserted into its correct location
    Temp ← NameList[ThisPointer]
    Pointer ← ThisPointer - 1

    WHILE (NameList[Pointer] > Temp) AND .....
        // move list item to next location
        NameList[.....] ← NameList[.....]
        Pointer ← .....
    ENDWHILE

    // insert value of Temp in correct location
    NameList[.....] ← .....

ENDFOR

```

[7]

- (ii) A special case is when NameList is already in order. The algorithm in part (a)(i) is applied to this special case.

Explain how many iterations are carried out for each of the loops.

.....

 [3]

- (b) An alternative sort algorithm is a bubble sort:

```
FOR ThisPointer ← 1 TO 9
    FOR Pointer ← 1 TO 9
        IF NameList[Pointer] > NameList[Pointer + 1]
            THEN
                Temp ← NameList[Pointer]
                NameList[Pointer] ← NameList[Pointer + 1]
                NameList[Pointer + 1] ← Temp
            ENDIF
        ENDFOR
    ENDFOR
```

- (i) As in part (a)(ii), a special case is when NameList is already in order. The algorithm in part (b) is applied to this special case.

Explain how many iterations are carried out for each of the loops.

.....
.....
.....
.....

[2]

- (ii) Rewrite the algorithm in part (b), using **pseudocode**, to reduce the number of unnecessary comparisons. Use the same variable names where appropriate.

[5]

- 6** A queue Abstract Data Type (ADT) has these associated operations:

- create queue
- add item to queue
- remove item from queue

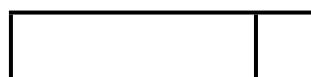
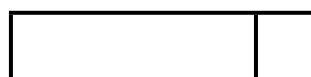
The queue ADT is to be implemented as a linked list of nodes.

Each node consists of data and a pointer to the next node.

- (a)** The following operations are carried out:

```
CreateQueue
AddName ("Ali")
AddName ("Jack")
AddName ("Ben")
AddName ("Ahmed")
RemoveName
AddName ("Jatinder")
RemoveName
```

Add appropriate labels to the diagram to show the final state of the queue. Use the space on the left as a workspace. Show your final answer in the node shapes on the right:



[3]

- (b) Using pseudocode, a record type, Node, is declared as follows:

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

The statement

```
DECLARE Queue : ARRAY[1:10] OF Node
```

reserves space for 10 nodes in array Queue.

- (i) The CreateQueue operation links all nodes and initialises the three pointers that need to be used: HeadPointer, TailPointer and FreePointer.

Complete the diagram to show the value of all pointers after CreateQueue has been executed.

Queue		
	Name	Pointer
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		
[10]		

HeadPointer

 TailPointer

 FreePointer

[4]

- (ii) The algorithm for adding a name to the queue is written, using pseudocode, as a procedure with the header:

```
PROCEDURE AddName (NewName)
```

where NewName is the new name to be added to the queue.

The procedure uses the variables as shown in the identifier table.

Identifier	Data type	Description
Queue	Array[1:10] OF Node	Array to store node data
NewName	STRING	Name to be added
FreePointer	INTEGER	Pointer to next free node in array
HeadPointer	INTEGER	Pointer to first node in queue
TailPointer	INTEGER	Pointer to last node in queue
CurrentPointer	INTEGER	Pointer to current node

```
PROCEDURE AddName (BYVALUE NewName : STRING)
    // Report error if no free nodes remaining
    IF FreePointer = 0
        THEN
            Report Error
    ELSE
        // new name placed in node at head of free list
        CurrentPointer ← FreePointer
        Queue[CurrentPointer].Name ← NewName
        // adjust free pointer
        FreePointer ← Queue[CurrentPointer].Pointer
        // if first name in queue then adjust head pointer
        IF HeadPointer = 0
            THEN
                HeadPointer ← CurrentPointer
        ENDIF
        // current node is new end of queue
        Queue[CurrentPointer].Pointer ← 0
        TailPointer ← CurrentPointer
    ENDIF
ENDPROCEDURE
```

Complete the **pseudocode** for the procedure RemoveName. Use the variables listed in the identifier table.

```
PROCEDURE RemoveName ()  
    // Report error if Queue is empty  
  
    .....  
  
    .....  
  
    .....  
  
    OUTPUT Queue[.....] .Name  
  
    // current node is head of queue  
  
    .....  
  
    // update head pointer  
  
    .....  
  
    // if only one element in queue then update tail pointer  
  
    .....  
  
    .....  
  
    .....  
  
    .....  
  
    // link released node to free list  
  
    .....  
  
    .....  
  
    .....  
  
ENDPROCEDURE
```

[6]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

May/June 2018

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **17** printed pages and **3** blank pages.

- 1 A declarative language is used to represent facts and rules about flights.

```

01 direct(edinburgh, paris).
02 direct(palma, rome).
03 direct(glasgow, palma).
04 direct(glasgow, vienna).
05 direct(glasgow, salzburg).
06
07 flies(paris, fly_jet).
08 flies(mumbai, british_air).
09 flies(palma, ciebe).
10 flies(vienna, fly_jet).
11 flies(salzburg, ciebe).
12
13 can_fly(X, Y) IF direct(X, Z) AND direct(Z, Y).

```

These clauses have the following meaning:

Clause	Explanation
01	There is a direct route from Edinburgh to Paris.
07	Fly Jet operates flights to Paris.
13	It is possible to fly from X to Y if there is a direct flight from X to Z and a direct flight from Z to Y.

- (a) More facts need to be included.

There is a direct flight from London to Rome and British Air flies to Rome.

14

15

[2]

- (b) Using the variable Q, the goal

flies(Q, fly_jet).

returns

Q = paris, vienna

Write the result returned by the goal

flies(K, ciebe).

K = [2]

- (c) Use the variable M to write the goal to find where you can fly direct from Glasgow.

..... [2]

- (d) If an airline flies to an airport, that airline also flies every direct route out of that airport.

Write a rule to represent this condition.

`flies(Y, X)`

IF

..... [3]

- (e) State what the following goal returns.

`can_fly(glasgow, rome).`

..... [1]

- 2 The array `ItemList[1:20]` stores data. A **bubble sort** sorts these data.

- (a) Complete the pseudocode algorithm for a bubble sort.

```

01  MaxIndex ← 20
02  NumberItems ← .....
03  FOR Outer ← 1 TO .....
04    FOR Inner ← 1 to NumberItems
05      IF ItemList[Inner] > .....
06        THEN
07          Temp ← ItemList[.....]
08          ItemList[Inner] ← ItemList[.....]
09          ItemList[Inner + 1] ← .....
10        ENDIF
11    ENDFOR
12    NumberItems ← .....
13  ENDFOR

```

[7]

- (b) The algorithm in part (a) is inefficient.

- (i) Explain why the algorithm in part (a) is inefficient.

.....
.....
.....
.....

[2]

- (ii) Explain how you would improve the efficiency of this algorithm.

.....
.....
.....
.....

[3]

- (c) An insertion sort is another sorting algorithm.

State **two** situations when an insertion sort is more efficient than a bubble sort. Give a reason for each.

Situation 1

.....

Reason

.....

.....

Situation 2

.....

Reason

.....

.....

[4]

- 3 An internet based music streaming service provides access to an unlimited number of songs for members to play.

The following pseudocode represents the operation of the service.

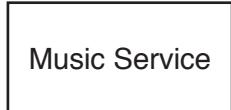
```

CALL OpenAccount()
CALL OperateAccount()
CALL CloseAccount()

PROCEDURE OperateAccount()
    WHILE RequestCloseAccount() = FALSE
        IF SubscriptionDue() = TRUE
            THEN
                CALL MakePayment()
            ELSE
                CALL PlaySong()
            ENDIF
        ENDWHILE
    ENDPROCEDURE

```

- (a) Complete the JSP structure diagram for this music service from the pseudocode given.



[5]

(b) The service needs extending so that members can download songs to play offline.

- When a member selects a song, the service checks if the song has already been downloaded.
- If the member has already downloaded the song, the member has the option to delete or play it.
- If the member has not already downloaded the song they have the option to download or stream it.

Complete the following JSP structure diagram to represent these new requirements.

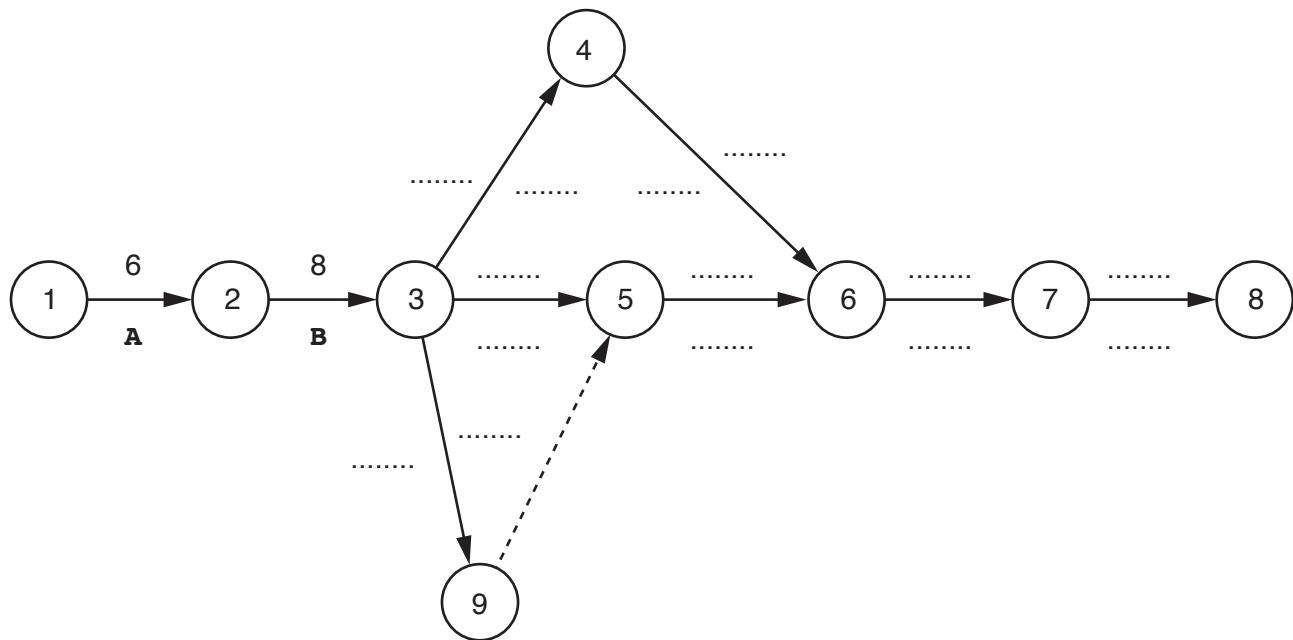


[4]

- 4 A software company is developing a new application. The project manager has created a work breakdown structure, as shown in the following table.

Activity		Days to complete	Predecessor
A	Gather user requirements	6	
B	Design work	8	A
C	Develop server code	4	B
D	Develop application code	5	B
E	User interface development	6	B
F	Test server code	2	C
G	Test application	2	D, E
H	Test application/server integration	5	F, G
I	Roll out mobile application	3	H

- (a) Use the data in the table to complete the following Program Evaluation Review Technique (PERT) chart.



[5]

- (b) Calculate the critical path (CP). State the:

activities that form the CP

duration of the CP

[2]

- (c) For activity F, state the:

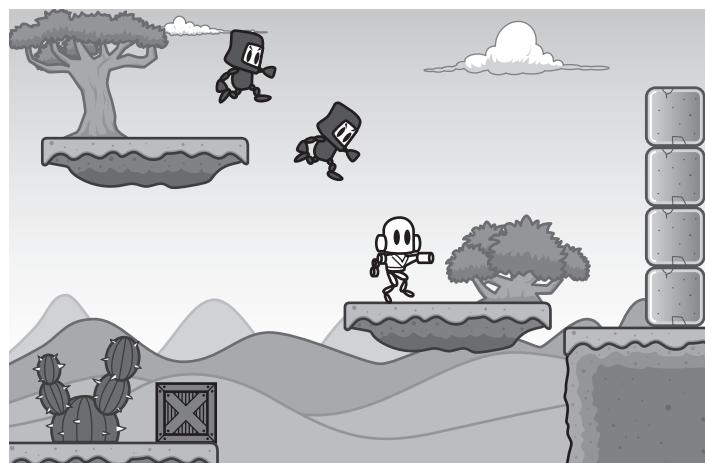
earliest start time

latest finish time

[2]

Question 5 begins on the next page.

- 5 A computer game is being developed using object-oriented programming. The following image is a screenshot from the game.



There are scenery elements and animated elements. The player's character is one of the animated elements.

Each game element has the attributes:

Attribute	Description	Example value
PositionX	The x coordinate of the game element.	92
PositionY	The y coordinate of the game element.	106
Width	The width of the game element.	150
Height	The height of the game element.	200
ImageFilename	The filename of the image file for the game element.	GameElementFrame1.png

Each game element has a method, `GetDetails()` that returns a string containing all the element's attributes.

The player's character is one of a number of animated elements. All animated elements have the attributes:

Attribute	Description	Example value
AnimationFrames	An array of GameElement	
Direction	A string giving the direction the object is travelling in.	"Left"
Strength	A value for the strength that indicates the power of the object.	2000
Health	A value for the health that indicates the health of the object.	100

The player's character can either move left or right, or jump.

- (a) Complete the following class diagram for the game.

You do not need to include any additional get or set methods.

GameElement
PositionX: INTEGER PositionY: INTEGER Width: INTEGER Height: INTEGER ImageFilename: STRING
Constructor() GetDetails()

AnimatedElement
AnimationFrames: ARRAY OF GameElement
.....
.....
.....
Constructor() AdjustHealth() AdjustStrength() DisplayAnimation()

Scenery
CauseDamage: BOOLEAN DamagePoints: INTEGER
Constructor() GiveDamagePoints()

Player
.....
.....
.....
.....
.....

[3]

- (b)** Write **program code** to define the GameElement class.

Programming language

Program code

- (c) The Scenery() class has two attributes, CauseDamage and DamagePoints.

If the attribute CauseDamage is TRUE, then the scenery element can cause damage.

The method `GiveDamagePoints()` checks whether the object can cause damage. If the object can cause damage, the method returns the integer value of the `DamagePoints` attribute.

Write program code for the Scenery class.

Programming language

Program code

.. [6]

(d) A new scenery object, `GiftBox`, is to be created.

(i) The attributes of `GiftBox` are as follows:

Attribute	Value
PositionX	150
PositionY	150
Width	50
Height	75
ImageFilename	"box.png"
CauseDamage	TRUE
DamagePoints	50

Write **program code** to create an instance of `GiftBox`.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

[3]

- (ii) An additional method, GetScenery(), returns all the attributes of the Scenery class.

Write program code for the GetScenery() method.

You should use the `GetDetails()` method that the `Scenery` class inherits from the `GameElement` class.

Programming language

Program code

[3]

... [3]

- 6 An Abstract Data Type (ADT) is used to create a linked list. The linked list is created as an array of records. The records are of type `ListNode`.

An example of a record of `ListNode` is shown in the following table.

Data Field	Value
Player	"Alvaro"
Pointer	1

- (a) (i) Use **pseudocode** to write a definition for the record type, `ListNode`.

.....

[3]

- (ii) An array, `Scorers`, will hold 10 nodes of type `ListNode`. Use **pseudocode** to write an array declaration for this array. The lower bound subscript is 0.

..... [2]

- (b) The linked list stores `ListNode` records in alphabetical order of player. The last node in the linked list always has a `Pointer` value of -1. The position of the first node in the linked list is held in the variable `ListHead`.

After some processing, the array and variables are in the state as follows:

Scorers		
ListHead	Player	Pointer
0	"Alvaro"	1
1	"Antoine"	3
2	"Dimitri"	7
3	"Cristiano"	2
4	"Gareth"	5
5	"Graziano"	6
6	"Olivier"	8
7	"Erik"	4
8	"Yaya"	9
9	"Zoto"	-1

A **recursive** function traverses the linked list to search for a player.

An example of calling the function, using pseudocode, is:

```
Position ← SearchList("Gareth", ListHead)
```

Complete the following **pseudocode** to implement the function `SearchList()`.

The function will return a value of 99 when a player is not found.

```
FUNCTION SearchList(Find : STRING, Position : INTEGER) RETURNS INTEGER
    IF Scorer[Position].Player = .....
        THEN
            RETURN .....
    ELSE
        IF Scorer[Position].Pointer <> -1
            THEN
                Position ← SearchList(Find, ....)
                RETURN .....
        ELSE
            RETURN .....
        ENDIF
    ENDIF
ENDFUNCTION
```

[5]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

May/June 2016

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **19** printed pages and **1** blank page.

- 1 A linked list abstract data type (ADT) is to be used to store and organise surnames.

This will be implemented with a 1D array and a start pointer. Elements of the array consist of a user-defined type. The user-defined type consists of a data value and a link pointer.

Identifier	Data type	Description
LinkedList	RECORD	User-defined type
Surname	STRING	Surname string
Ptr	INTEGER	Link pointers for the linked list

- (a) (i) Write **pseudocode** to declare the type `LinkedList`.

.....
.....
.....
.....

[3]

- (ii) The 1D array is implemented with an array `SurnameList` of type `LinkedList`.

Write the **pseudocode** declaration statement for `SurnameList`. The lower and upper bounds of the array are 1 and 5000 respectively.

..... [2]

- (b) The following surnames are organised as a linked list with a start pointer `StartPtr`.

`StartPtr: 3`

	1	2	3	4	5	6	5000
Surname	Liu	Yang	Chan	Wu	Zhao	Huang	...	
Ptr	4	5	6	2	0	1	...	

State the value of the following:

- (i) `SurnameList[4].Surname` [1]

- (ii) `SurnameList[StartPtr].Ptr` [1]

- (c) Pseudocode is to be written to search the linked list for a surname input by the user.

Identifier	Data type	Description
ThisSurname	STRING	The surname to search for
Current	INTEGER	Index to array SurnameList
StartPtr	INTEGER	Index to array SurnameList. Points to the element at the start of the linked list

- (i) Study the pseudocode in part (c)(ii).

Complete the table above by adding the missing identifier details.

[2]

- (ii) Complete the pseudocode.

```

01 Current ← .....
02 IF Current = 0
03   THEN
04     OUTPUT .....
05 ELSE
06   IsFound ← .....
07   INPUT ThisSurname
08 REPEAT
09   IF ..... = ThisSurname
10     THEN
11       IsFound ← TRUE
12       OUTPUT "Surname found at position ", Current
13     ELSE
14       // move to the next list item
15       .....
16   ENDIF
17 UNTIL IsFound = TRUE OR .....
18 IF IsFound = FALSE
19   THEN
20   OUTPUT "Not Found"
21 ENDIF
22 ENDIF

```

[6]

- 2 (a) (i) State what is meant by a recursively defined procedure.

.....
.....

[1]

- (ii) Write the line number from the pseudocode shown in part (b) that shows the procedure X is recursive.

[1]

- (b) The recursive procedure X is defined as follows:

```

01 PROCEDURE X(Index, Item)
02     IF MyList[Index] > 0
03         THEN
04             IF MyList[Index] >= Item
05                 THEN
06                     MyList[Index] ← MyList[Index + 1]
07                 ENDIF
08             CALL X(Index + 1, Item)
09         ENDIF
10     ENDPROCEDURE

```

An array MyList is used to store a sorted data set of non-zero integers. Unused cells contain zero.

	1	2	3	4	5	6	7	8	9	10
MyList	3	5	8	9	13	16	27	0	0	0

- (i) Complete the trace table for the dry-run of the pseudocode for the procedure
CALL X(1, 9).

		MyList									
Index	Item	1	2	3	4	5	6	7	8	9	10
1	9	3	5	8	9	13	16	27	0	0	0

[4]

- (ii) State the purpose of procedure X when used with the array MyList.

.....
.....

[1]

- 3 A car hire company hires cars to customers. Each time a car is hired, this is treated as a transaction.

For each transaction, the following data are stored.

For the customer:

- customer name
- ID number

For the hire:

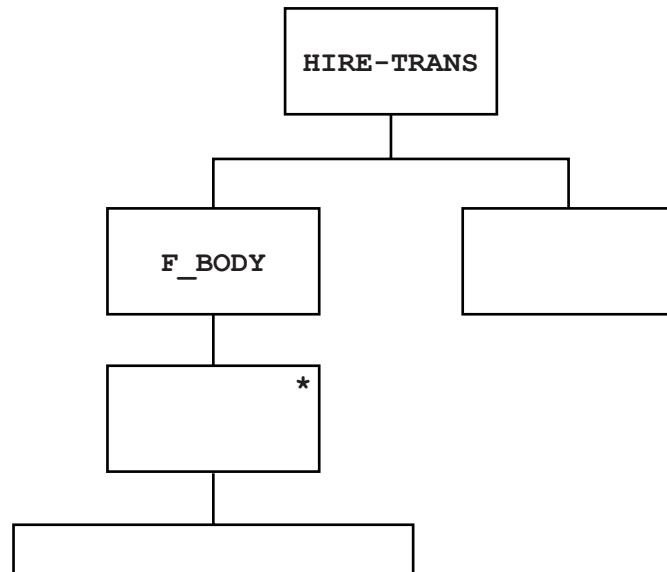
- car registration
- hire start date
- number of days hired

The transaction data are stored in a text file HIRE-TRANS. The file is made up of a file body, F_BODY, and a file trailer, F_TRAILER.

F_BODY has one transaction, TRANS, on each line.

- (a) The first step in Jackson Structured Programming (JSP) design is to produce a JSP data structure diagram.

Complete the following JSP data structure diagram.



- (b) The computer system will produce many printed reports.

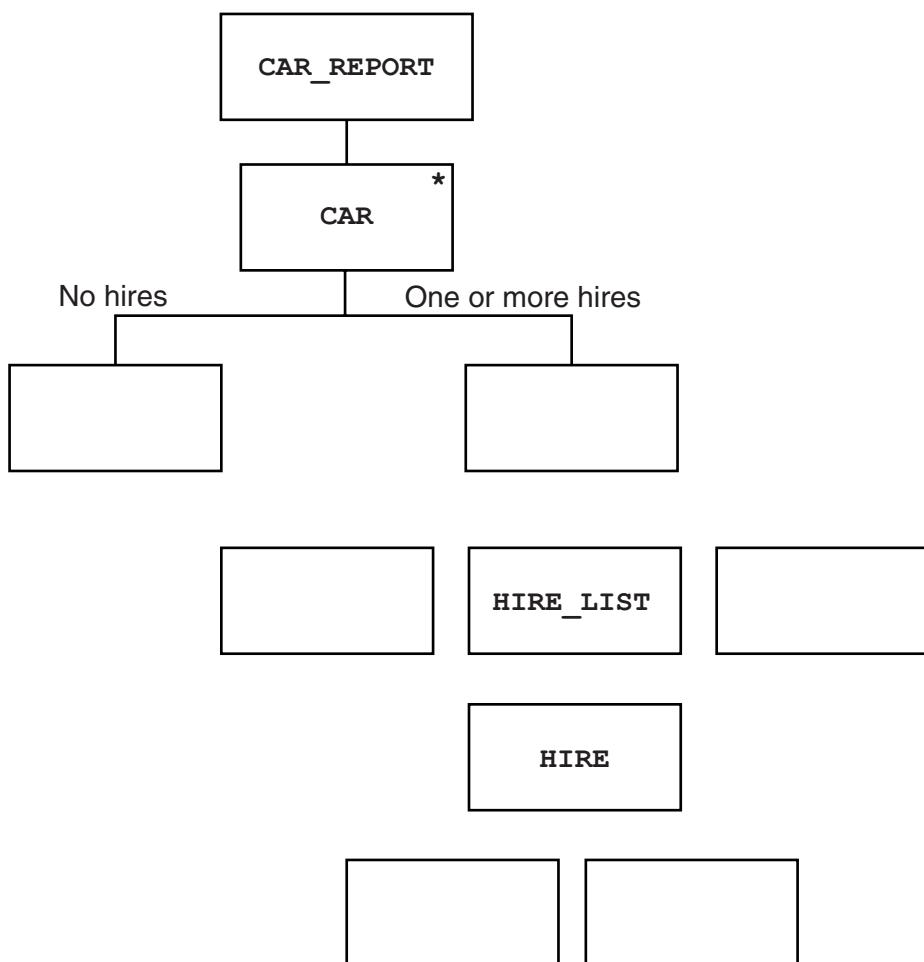
One report is CAR_REPORT. This displays all hire data for all cars.

For each car, the following data are displayed:

- the car data
- a list of all the hires
- the total number of hires

A car with zero hires is not included on the report.

Complete the following CAR_REPORT JSP data structure diagram.



[5]

- 4 When a car reaches a certain age, a safety assessment has to be carried out. A car's brakes and tyres must be tested. The tyre test result and the brakes test result for each car are recorded. If the car passes the assessment, a safety certificate is issued.

Cars have a unique three-character registration.

The following knowledge base is used:

```

01 car(a05).
02 car(h04).
03 car(a03).
04 car(h07).
05 car(a23).
06 car(p05).
07 car(b04).
08 carRegYear(a05, 2015).
09 carRegYear(h04, 2013).
10 carRegYear(a03, 2008).
11 carRegYear(h07, 2011).
12 carRegYear(a23, 2008).
13 carRegYear(p05, 2014).
14 carRegYear(b04, 2014).
15 testBrakes(h07, pass).
16 testTyres(h07, fail).
17 testBrakes(a03, fail).
18 testTyres(a03, fail).
19 testBrakes(a23, pass).
20 testTyres(a23, pass).
21 carAssessmentDue if carRegYear(Car, RegYear)
                           and RegYear <= DeadlineYear.
22 issueCertificate(Car) if testTyres(Car, Result) and
                           testBrakes(Car, Result) and Result = pass.

```

- (a) (i) DeadlineYear is assigned value 2011.

Identify the car registrations for cars which are due to be tested.

..... [1]

- (ii) State how clause 22 determines whether or not a safety certificate will be issued.

..... [1]

- (b)** If a car fails one of the two tests, a retest is allowed.

Write a new rule for this.

retestAllowed(.....) if

.....

..... [3]

- (c)** Logic programming uses a data structure called a list.

A new fact is added to the knowledge base.

23 carList = [a03, p05, b04, h04, h07, a23].

The following notation and operators are to be used with a list:

[X | Y] denotes a list with:

- X the first list element
- Y the list consisting of the remaining list elements

[] denotes an empty list

- (i)** The list [a07, p03] is denoted by [A | B]

State the value of A and B.

A =

B = [2]

- (ii)** The lists [c03, d02, n05 | C] and [c03, d02, n05, p05, m04] are identical.

State the value of C.

C = [1]

- (iii)** The list [a06, a02] is denoted by [D, E | F]

State the value of F.

F = [1]

- (d) The predicate `conCatCompare` is defined as a rule and returns TRUE or FALSE as follows:

```
conCatCompare(X, Y, Z)
```

Concatenates the lists X and Y and compares the new list with list Z.

If equal, the clause evaluates to TRUE, otherwise FALSE.

Consider the clause:

```
conCatCompare(X, Y, [a7,b6,c4])
```

If:

- the clause evaluates to TRUE
- and Y represents the list [a7, b6, c4]

State the value of X.

X = [1]

- 5 (a) A program calculates the exam grade awarded from a mark input by the user. The code is written as a function CalculateGrade.

The function:

- has a single parameter **Mark** of **INTEGER** data type
- returns the grade awarded **Grade** of **STRING** data type

The logic for calculating the grade is as follows:

Mark	Grade
Under 40	FAIL
40 and over and under 55	PASS
55 and over and under 70	MERIT
70 and over	DISTINCTION

The programmer designs the following table for test data:

Mark	Description	Expected result (Grade)
	Normal	
	Abnormal	
	Extreme/Boundary	

- (i) Complete the table above. [3]
- (ii) State why this table design is suitable for black box testing.

..... [1]

(b) When designing and writing program code, explain what is meant by:

- an exception
- exception handling

.....
.....
.....
.....
.....
.....

[3]

(c) A program is to be written to read a list of exam marks from an existing text file into a 1D array.

Each line of the file stores the mark for one student.

State **three** exceptions that a programmer should anticipate for this program.

1

.....

2

.....

3

.....

[3]

- (d) The following pseudocode is to read two numbers:

```

01  DECLARE Num1      : INTEGER
02  DECLARE Num2      : INTEGER
03  DECLARE Answer : INTEGER
04  TRY
05      OUTPUT "First number..."
06      INPUT Num1
07      OUTPUT "Second number..."
08      INPUT Num2
09      Answer ← Num1 / (Num2 - 6)
10      OUTPUT Answer
11  EXCEPT ThisException : EXCEPTION
12      OUTPUT ThisException.Message
13  FINALLY
14      // remainder of the program follows
...
29
30 ENDTRY

```



The programmer writes the corresponding program code.

A user inputs the number 53 followed by 6. The following output is produced:

```

First number...53
Second number...6
Arithmetic operation resulted in an overflow

```

- (i) State the pseudocode line number which causes the exception to be raised.

.....

[1]

- (ii) Explain the purpose of the pseudocode on lines 11 and 12.

.....
.....
.....
.....
.....
.....

[3]

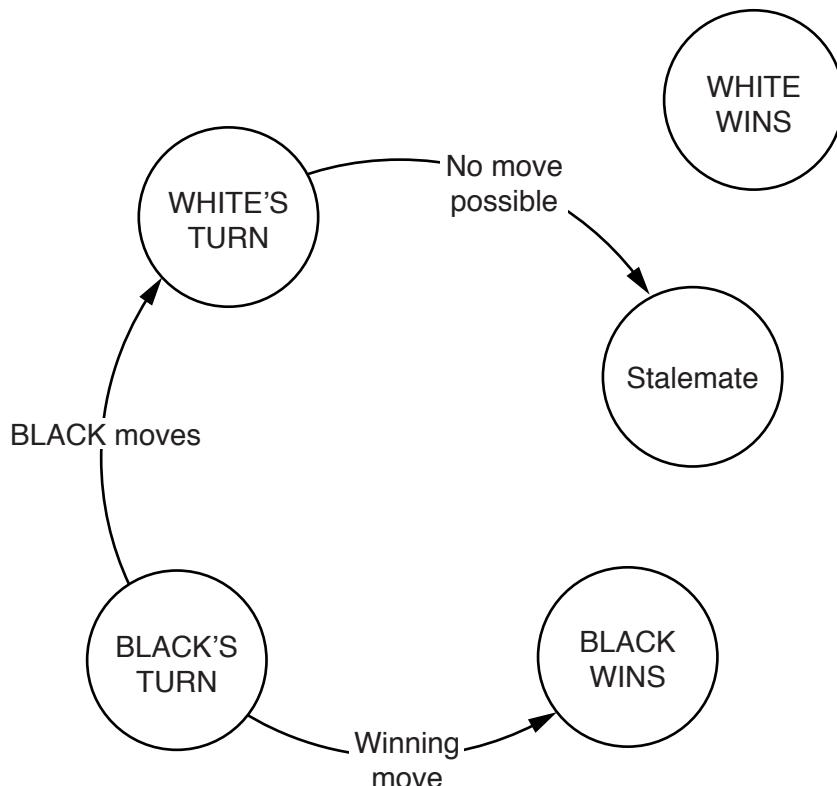
- 6 In a board game, one player has white pieces and the other player has black pieces. Players take alternate turns to move one of their pieces. White always makes the first move.

The game ends if:

- a player is unable to make a move when it is their turn. In this case, there is no winner. This is called 'stalemate'.
- a player wins the game as a result of their last move and is called a 'winner'.

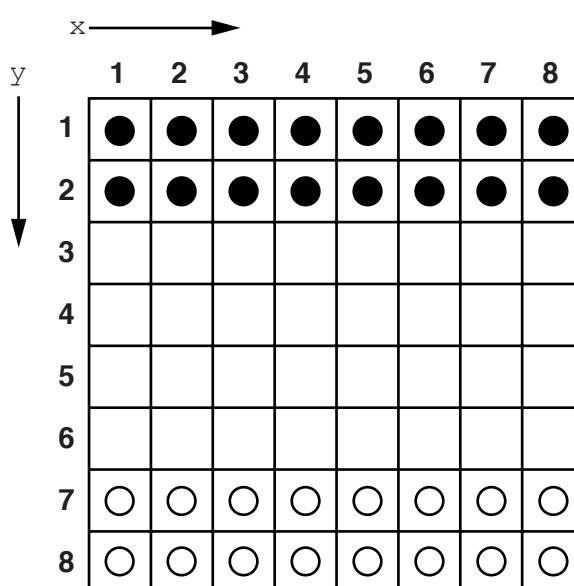
- (a) A state-transition diagram is drawn to clarify how the game is played.

Complete the following state-transition diagram.



[4]

- (b) The layout of the board at the start of the game is shown below:



The programmer decides to use a 2D array to represent the board. The index numbering to be used is as shown.

Each square on the board is either occupied by one piece only, or is empty.

The data stored in the array indicate whether or not that square is occupied, and if so, with a black piece or a white piece.

- (i) Write **program code** to initialise the contents of the array to represent the board at the start of the game. Use characters as follows for each square:

- 'B' represents a black piece 
 - 'W' represents a white piece 
 - 'E' represents an empty square

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language

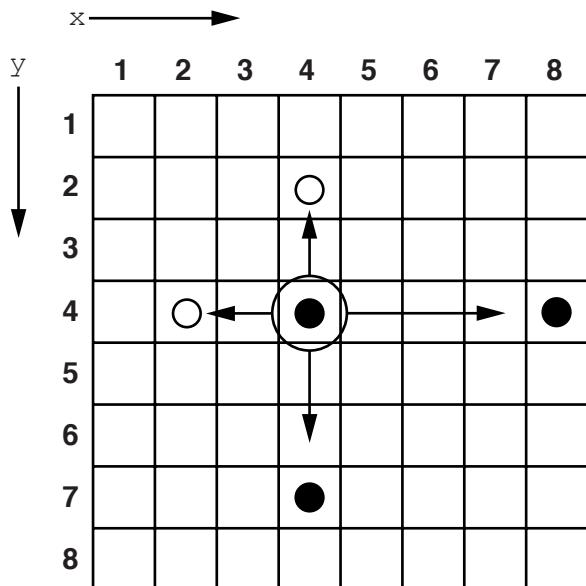
- (ii) When a piece is to be moved, a procedure will calculate and output the possible destination squares for the moving piece.

A piece can move one or more squares, in the x or y direction, from its current position.

This will be a move:

- either to an empty square, with no occupied squares on the way
- or to a square containing a piece belonging to another player, with no occupied squares on the way. The other player's piece is then removed.

For example, for the circled black piece there are nine possible destination squares. Each of the two destination squares contains a white piece which would be removed.



The program requires a procedure `ValidMoves`.

It needs three parameters:

- `PieceColour` – colour of the moving piece
- `xCurrent` – current x position
- `yCurrent` – current y position

The procedure will calculate all possible destination squares **in the x direction only**.

Example output for the circled black piece is:

```

Possible moves are:
Moving LEFT
3 4
2 4 REMOVE piece
Moving RIGHT
5 4
6 4
7 4

```

Write **program code** for procedure `ValidMoves` with the following procedure header:

```

PROCEDURE ValidMoves(PieceColour : CHAR, xCurrent : INTEGER,
                      yCurrent : INTEGER).

```

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language

. [5]

- (c) The problem is well suited to an object-oriented design followed by object-oriented programming.

- (i) Describe how classes and objects could be used in this problem.

.....
.....
.....
.....
.....

[2]

- (ii) For a class you identified in part(c)(i), state **two** properties and **two** methods.

Class

Properties

1
2

Methods

1,
2

[2]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/41

Paper 4 Practical

May/June 2021

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the document **evidence.doc**.

Make sure that your name, centre number and candidate number will appear on every page of this document. This document will contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

A list is an alternative to an array.

A source file is used to answer question 3. The file is called **TreasureChestData.txt**

- 1 An unordered linked list uses a 1D array to store the data.

Each item in the linked list is of a record type, node, with a field data and a field nextNode.

The current contents of the linked list are:

startPointer	0	Index	data	nextNode
emptyList	5	0	1	1
		1	5	4
		2	6	7
		3	7	-1
		4	2	2
		5	0	6
		6	0	8
		7	56	3
		8	0	9
		9	0	-1

- (a) The following is pseudocode for the record type node.

```

TYPE node
  DECLARE data : INTEGER
  DECLARE nextNode : INTEGER
ENDTYPE

```

Write program code to declare the record type node.

Save your program as **question1**.

Copy and paste the program code into **part 1(a)** in the evidence document.

- (b) Write program code for the main program.

Declare a 1D array of type `node` with the identifier `linkedList`, and initialise it with the data shown in the table on page 2. Declare the pointers.

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[4]

- (c) The procedure `outputNodes()` takes the array and `startPointer` as parameters. The procedure outputs the data from the linked list by following the `nextNode` values.

- (i) Write program code for the procedure `outputNodes()`.

Save your program.

Copy and paste the program code into **part 1(c)(i)** in the evidence document.

[6]

- (ii) Edit the main program to call the procedure `outputNodes()`.

Take a screenshot to show the output of the procedure `outputNodes()`.

Save your program.

Copy and paste the screenshot into **part 1(c)(ii)** in the evidence document.

[1]

- (d) The function, `addNode()`, takes the linked list and pointers as parameters, then takes as input the data to be added to the end of the `linkedList`.

The function adds the node in the next available space, updates the pointers and returns True. If there are no empty nodes, it returns False.

- (i) Write program code for the function `addNode()`.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[7]

- (ii) Edit the main program to:

- call `addNode()`
- output an appropriate message depending on the result returned from `addNode()`
- call `outputNodes()` twice; once before calling `addNode()` and once after calling `addNode()`.

Save your program.

Copy and paste the program code into **part 1(d)(ii)** in the evidence document.

[3]

- (iii) Test your program by inputting the data value 5 and take a screenshot to show the output.

Save your program.

Copy and paste the screenshot into **part 1(d)(iii)** in the evidence document.

[1]

BLANK PAGE

2 A program stores the following ten integers in a 1D array with the identifier `arrayData`.

10 5 6 7 1 12 13 15 21 8

(a) Write program code for a **new program** to:

- declare the global 1D array, `arrayData`, with ten elements
- initialise `arrayData` in the main program using the data values shown.

Save your program as **question2**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[2]

(b) (i) A function, `linearSearch()`, takes an integer as a parameter and performs a linear search on `arrayData` to find the parameter value. It returns `True` if it was found and `False` if it was not found.

Write program code for the function `linearSearch()`.

Save your program.

Copy and paste the program code into **part 2(b)(i)** in the evidence document.

[6]

(ii) Edit the main program to:

- allow the user to input an integer value
- pass the value to `linearSearch()` as the parameter
- output an appropriate message to tell the user whether the search value was found or not.

Save your program.

Copy and paste the program code into **part 2(b)(ii)** in the evidence document.

[4]

(iii) Test your program with one value that is in the array and one value that is not in the array.

Take a screenshot to show the result of each test.

Save your program.

Copy and paste the screenshots into **part 2(b)(iii)** in the evidence document.

[2]

- (c) The following bubble sort pseudocode algorithm sorts the data in `theArray` into descending numerical order. There are **five** incomplete statements.

```
PROCEDURE bubbleSort()

    DECLARE temp : INTEGER

    FOR x ← 0 to .....
        FOR y ← 0 to .....
            IF theArray[y] ..... theArray[y + 1] THEN
                temp ← theArray[y]
                theArray[y] ← .....
                theArray[y + 1] ← .....
            ENDIF
        NEXT y
    NEXT x
ENDPROCEDURE
```

Write program code for the procedure `bubbleSort()` to sort the data in `arrayData` into descending order.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[6]

- 3 A computer game requires users to travel around a world to find and open treasure chests. Each treasure chest has a mathematics question inside. The user enters the answer. The number of points awarded depends on the number of attempts before the user gives the correct answer.

The program will be created using object-oriented programming (OOP).

The following class diagram describes the class `TreasureChest`.

TreasureChest	
question : STRING answer : INTEGER points : INTEGER	// stores the question // stores the answer // stores the maximum possible number of points available for this chest
constructor()	// takes question, answer and points as parameters and creates an instance of an object
getQuestion()	// returns the question
checkAnswer()	// takes the user's answer as a parameter and returns True if it is correct, otherwise returns False
getPoints()	// takes the number of attempts as a parameter and returns the number of points awarded

- (a) Create a new program.

Write program code to declare the class `TreasureChest`.

Do **not** write any other methods.

The attributes are private.

If you are using the Python programming language, include attribute declarations using comments.

Save your program as **question3**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[5]

- (b) The text file `TreasureChestData.txt` stores data for five questions, in the order of question, answer, points.

For example, the first three lines of the file are for the first question:

```
2*2 question  
4 answer  
10 points
```

Write program code for the procedure, `readData()` to:

- read each question, answer and points from the text file
- create an object of type `TreasureChest` for each question
- declare an array named `arrayTreasure` of type `TreasureChest`
- append each object to the array
- use exception handling to output an appropriate message if the file is not found.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[8]

- (c) The main program repeats each question until the user inputs the correct answer. The number of points awarded depends on the number of attempts before the user gives the correct answer.

- (i) The class `TreasureChest` has a method `getQuestion()` that returns the question.

Write the method `getQuestion()`.

Save your program.

Copy and paste the program code into **part 3(c)(i)** in the evidence document.

[1]

- (ii) The class `TreasureChest` has a method `checkAnswer()` that takes the user's answer as a parameter. It returns `True` if the answer is correct and `False` otherwise.

Write the method `checkAnswer()`.

Save your program.

Copy and paste the program code into **part 3(c)(ii)** in the evidence document.

[3]

- (iii) The class `TreasureChest` has a method `getPoints()` that takes the number of attempts as a parameter.

- If the number of attempts is 1, it returns the value of `points`.
- If the number of attempts is 2, it returns the integer value of `points` divided by 2 (`DIV 2`).
- If the number of attempts is 3 or 4, it returns the integer value of `points` divided by 4 (`DIV 4`).
- If the number of attempts is not 1 or 2 or 3 or 4, it returns 0 (zero).

For example, a question is worth 100 points and the user took 2 attempts to give the correct answer. The user is awarded 50 points ($100 \text{ DIV } 2$).

Write the method `getPoints()`.

Save your program.

Copy and paste the program code into **part 3(c)(iii)** in the evidence document.

[5]

(iv) Write program code for the main program to:

- call the procedure `readData()`
- ask the user to enter a question number between 1 and 5
- output the question that matches the question number entered by the user
- check if the answer input by the user is correct using the method `checkAnswer()`
- repeat the question until the user inputs the correct answer
- count how many times the user attempted the question
- use the method `getPoints()` to return the number of points awarded
- output the number of points the user is awarded.

Save your program.

Copy and paste the program code into **part 3(c)(iv)** in the evidence document.

[7]

(v) Test the program.

Take a screenshot showing the input(s) and output(s) for each of the following two tests.

In the first test:

- select question 1 and answer it correctly the first time.

In the second test:

- select question 5 and answer it correctly the second time.

Save your program.

Copy and paste the screenshots into **part 3(c)(v)** in the evidence document.

[2]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

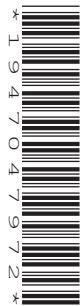
COMPUTER SCIENCE

9618/43

Paper 4 Practical

May/June 2023

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the evidence document **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

One source file is used to answer **Question 1** and two source files are used to answer **Question 3**. The files are called **Data.txt**, **AnimalData.txt** and **ColourData.txt**

A class declaration can be used to declare a record.

A list is an alternative to an array.

1 A program reads data from a file and searches for specific data.

- (a) The main program needs to read 25 integer data items from the text file **Data.txt** into a local 1D array, **DataArray**
 - (i) Write program code to declare the local array **DataArray**

Save your program as **Question1_J2023**.

Copy and paste the program code into **part 1(a)(i)** in the evidence document.

[1]

- (ii) Amend the main program to read the contents of **Data.txt** into **DataArray**

Save your program.

Copy and paste the program code into **part 1(a)(ii)** in the evidence document.

[4]

- (b) (i) The procedure **PrintArray()** takes an integer array as a parameter and outputs the contents of the array in the order they are stored.

The items are printed on the same line, for example:

10 4 5 13 25

Write program code for the procedure **PrintArray()**

Save your program.

Copy and paste the program code into **part 1(b)(i)** in the evidence document.

[3]

- (ii) Amend the main program to output the contents of `DataArray` using the procedure `PrintArray()`

Save your program.

Copy and paste the program code into **part 1(b)(ii)** in the evidence document.

[1]

- (iii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 1(b)(iii)** in the evidence document.

[1]

- (c) The function `LinearSearch()`:

- takes an integer array and integer search value as parameters
- counts and returns the number of times the search value is found in the array.

Write program code for the function `LinearSearch()`

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[3]

- (d) (i) Amend the main program to:

- prompt the user to input a whole number between 0 and 100 inclusive
- read and validate the input from the user
- call `LinearSearch()` with `DataArray` and the validated input value
- output the result in the format:

The number 7 is found 2 times.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[4]

- (ii) Test your program by inputting the number 12.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 1(d)(ii)** in the evidence document.

[1]

- 2 A computer game is being designed that will include different vehicles. A prototype for the game is being developed using object-oriented programming.

The class `Vehicle` stores data about the vehicles. Each vehicle has an identification name, a maximum speed, a current speed and a horizontal position. The value `IncreaseAmount` is added to the current speed each time the vehicle increases its speed.

Vehicle	
<code>ID : STRING</code>	stores the identification name for the vehicle
<code>MaxSpeed : INTEGER</code>	stores the maximum speed
<code>CurrentSpeed : INTEGER</code>	stores the current speed
<code>IncreaseAmount : INTEGER</code>	stores the amount <code>CurrentSpeed</code> increases by
<code>HorizontalPosition : INTEGER</code>	stores the horizontal position
<code>Constructor()</code>	initialises <code>ID</code> , <code>MaxSpeed</code> and <code>IncreaseAmount</code> to the parameter values initialises both <code>CurrentSpeed</code> and <code>HorizontalPosition</code> to 0
<code>GetCurrentSpeed()</code>	returns the current speed
<code>GetIncreaseAmount()</code>	returns the increase amount
<code>GetHorizontalPosition()</code>	returns the horizontal position
<code>GetMaxSpeed()</code>	returns the maximum speed
<code>SetCurrentSpeed()</code>	assigns the parameter to the current speed
<code>SetHorizontalPosition()</code>	assigns the parameter to the horizontal position
<code>IncreaseSpeed()</code>	calculates and stores the new speed and horizontal position of the vehicle

- (a) (i) Write program code to declare the class `Vehicle`. All attributes must be private.

You only need to declare the class and its constructor. Do not declare any other methods.

Use your programming language's appropriate constructor.

If you are writing program code in Python, include attribute declarations using comments.

Save your program as **Question2_J2023**.

Copy and paste the program code into **part 2(a)(i)** in the evidence document.

[5]

- (ii) Write program code for the get methods `GetCurrentSpeed()`, `GetIncreaseAmount()`, `GetMaxSpeed()` and `GetHorizontalPosition()`

Save your program.

Copy and paste the program code into **part 2(a)(ii)** in the evidence document.

[3]

- (iii) Write program code for the set methods `SetCurrentSpeed()` and `SetHorizontalPosition()`

Save your program.

Copy and paste the program code into **part 2(a)(iii)** in the evidence document.

[3]

- (iv) The method `IncreaseSpeed()`:

- adds `IncreaseAmount` to the current speed
- adds the updated current speed to the horizontal position.

The current speed of a vehicle cannot exceed its maximum speed.

Write program code for the method `IncreaseSpeed()`

Save your program.

Copy and paste the program code into **part 2(a)(iv)** in the evidence document.

[3]

- (b) The child class `Helicopter` inherits from the parent class `Vehicle`. A helicopter also has a vertical position and changes the vertical position when it increases speed.

Helicopter	
<code>VerticalPosition : INTEGER</code>	stores the vertical position
<code>VerticalChange : INTEGER</code>	stores the amount <code>VerticalPosition</code> changes by
<code>MaxHeight : INTEGER</code>	stores the maximum height the helicopter can reach
<code>Constructor()</code>	takes the ID, maximum speed, increase amount, vertical change and maximum height as parameters initialises the vertical position to 0
<code>GetVerticalPosition()</code>	returns the vertical position
<code>IncreaseSpeed()</code>	changes the current speed, horizontal and vertical position of the helicopter

- (i) Write program code to declare the class `Helicopter`. You only need to declare the class and its constructor. You do not need to declare the other methods.

Use your programming language's appropriate constructor.

All attributes must be private.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into **part 2(b)(i)** in the evidence document.

[5]

- (ii) The `Helicopter` method `IncreaseSpeed()` overrides the method from the parent class and:

- adds the amount of vertical change to the vertical position
- adds `IncreaseAmount` to the current speed
- adds the updated current speed to the horizontal position.

The vertical position of a helicopter cannot exceed its maximum height.

The current speed of a helicopter cannot exceed its maximum speed.

Write program code for the method `IncreaseSpeed()`

Save your program.

Copy and paste the program code into **part 2(b)(ii)** in the evidence document.

[4]

- (c) A procedure needs to output the horizontal position and speed of a vehicle. If the vehicle is a helicopter, it also outputs the vertical position.

All outputs must include appropriate messages.

Write program code for this procedure.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[3]

- (d) The main program needs to:

- instantiate a car as a new vehicle with the ID "Tiger", a maximum speed of 100 and an increase amount of 20
- instantiate a new helicopter with the ID "Lion", a maximum speed of 350, an increase amount of 40, a vertical change of 3 and a maximum height of 100
- call `IncreaseSpeed()` twice for the car and then call the output procedure from **part 2(c)** for the car
- call `IncreaseSpeed()` twice for the helicopter and then call the output procedure from **part 2(c)** for the helicopter.

- (i) Write program code for the main program.

Save your program.

Copy and paste the program code into **part 2(d)(i)** in the evidence document.

[5]

- (ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 2(d)(ii)** in the evidence document.

[1]

- 3 A program implements two stacks using 1D arrays. One stack stores the names of colours. One stack stores the names of animals.

(a) The program contains the following global arrays and variables:

- 1D array `Animal` to store the names of up to 20 animals.
- 1D array `Colour` to store the names of up to 10 colours.
- `AnimalTopPointer` to point to the next free space in the array `Animal`, initialised to 0.
- `ColourTopPointer` to point to the next free space in the array `Colour`, initialised to 0.

Write program code to declare the global arrays and variables.

Save your program as **Question3_J2023**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[3]

(b) (i) Study the pseudocode function `PushAnimal()`:

```

FUNCTION PushAnimal(DataToPush : STRING) RETURNS BOOLEAN

IF AnimalTopPointer = 20 THEN
    RETURN FALSE
ELSE
    Animal[AnimalTopPointer] ← DataToPush
    AnimalTopPointer ← AnimalTopPointer + 1
    RETURN TRUE
ENDIF

ENDFUNCTION

```

Write program code for the function `PushAnimal()`

Save your program.

Copy and paste the program code into **part 3(b)(i)** in the evidence document.

[3]

(ii) Study the pseudocode function `PopAnimal()`:

```

FUNCTION PopAnimal() RETURNS STRING

    DECLARE ReturnData : STRING

    IF AnimalTopPointer = 0 THEN

        RETURN ""

    ELSE

        ReturnData ← Animal[AnimalTopPointer - 1]

        AnimalTopPointer ← AnimalTopPointer - 1

        RETURN ReturnData

    ENDIF

ENDFUNCTION

```

Write program code to declare the function `PopAnimal()`

Save your program.

Copy and paste the program code into **part 3(b)(ii)** in the evidence document.

[3]

(iii) The procedure `ReadData()`:

- reads the animal names from the file `AnimalData.txt`
- uses `PushAnimal()` to insert each name onto the stack
- uses appropriate exception handling if the file does not exist.

Write program code for the procedure `ReadData()`

Save your program.

Copy and paste the program code into **part 3(b)(iii)** in the evidence document.

[5]

(iv) The function `PushColour()` performs the same actions as `PushAnimal()` but inserts an item into `Colour`.

The function `PopColour()` performs the same actions as `PopAnimal()` but removes the next item from `Colour`.

Write program code for the functions `PushColour()` and `PopColour()`

Save your program.

Copy and paste the program code into **part 3(b)(iv)** in the evidence document.

[2]

[Turn over

(v) Amend the procedure `ReadData()` so that it also:

- reads the colours from the text file `ColourData.txt`
- uses `PushColour()` to insert each colour onto the stack
- uses appropriate exception handling if the file does not exist.

Save your program.

Copy and paste the program code into **part 3(b)(v)** in the evidence document.

[2]

(c) The procedure `OutputItem()`:

- pops the next item from both `Animal` and `Colour`
- outputs the colour and animal on one line, for example "black horse"

If there is no data in `Colour`:

- the animal is pushed back onto `Animal`
- "No colour" is output.

If there is no data in `Animal`:

- the colour is pushed back onto `Colour`
- "No animal" is output.

Write program code for the procedure `OutputItem()`

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[5]

(d) The main program:

- calls the procedure `ReadData()`
- calls `OutputItem()` **four times**.

(i) Write program code for the main program.

Save your program.

Copy and paste the program code into **part 3(d)(i)** in the evidence document.

[1]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into **part 3(d)(ii)** in the evidence document.

[1]

BLANK PAGE

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.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

October/November 2017

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

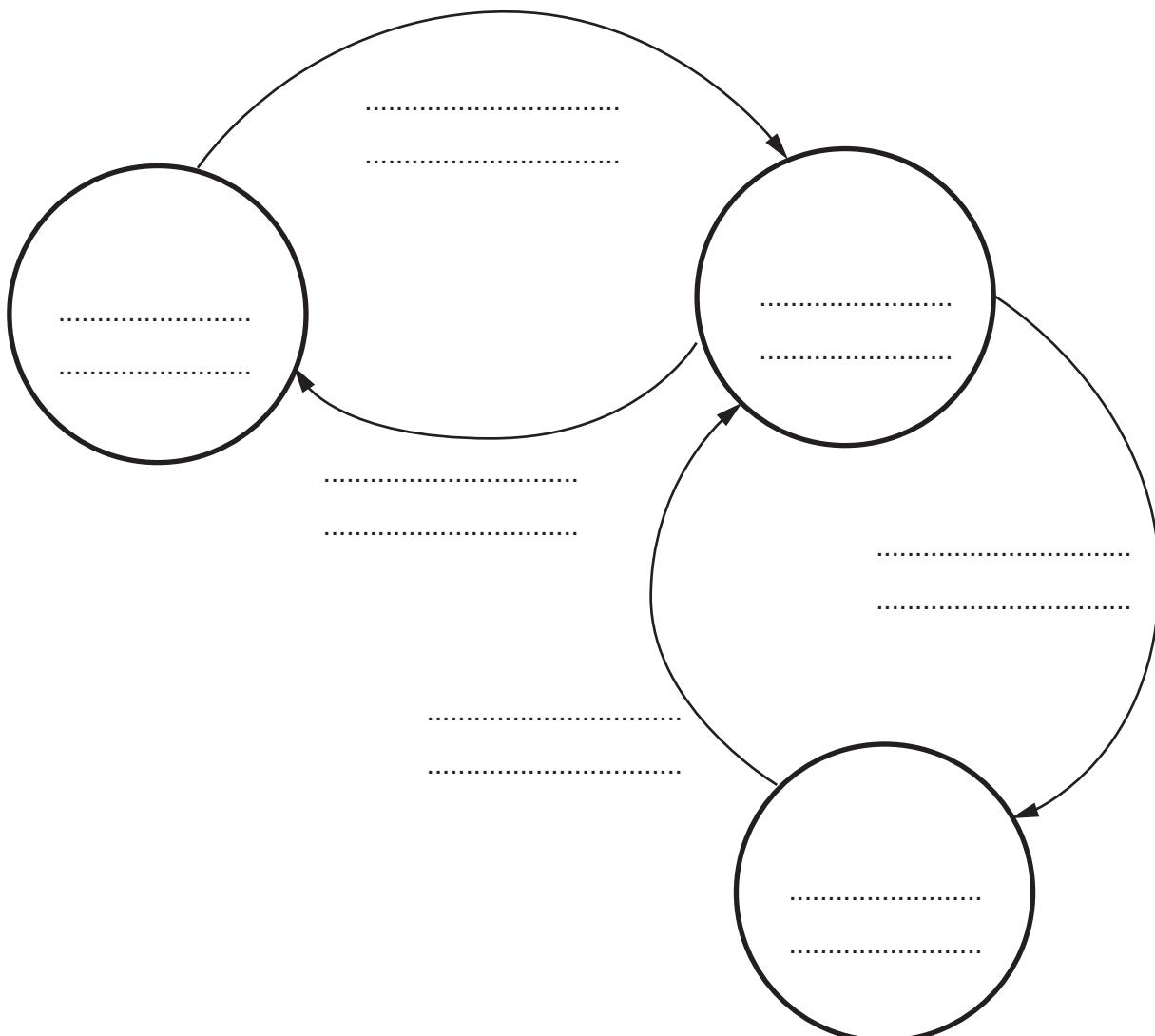
- 1 A greenhouse has a window that automatically opens and closes depending on the internal temperature.

If the temperature rises above 20°C , the window half opens. If the temperature rises above 30°C , the window fully opens. If the temperature drops below 25°C , the window returns to being half open. If the temperature drops below 15°C , the window fully closes.

The window has three possible states: **Closed**, **Half Open** and **Fully Open**.

Current state	Event	Next state
Closed	Temperature rises above 20°C	Half Open
Half Open	Temperature drops below 15°C	Closed
Half Open	Temperature rises above 30°C	Fully Open
Fully Open	Temperature drops below 25°C	Half Open

Complete the state-transition diagram for the window:



[7]

- 2 (a) (i) State how repetition is shown in a Jackson Structured Programming (JSP) structure diagram.

.....
..... [1]

- (ii) State how selection is shown in a JSP structure diagram.

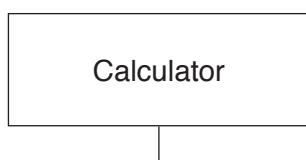
.....
..... [1]

- (b) A simple calculator is to be created.

The calculator is to be used as follows:

- User inputs 2 numbers (x and y).
- User inputs an operator (+, -, * or /).
- The calculator computes the answer.
- The calculator displays the answer.

Draw a JSP diagram for the calculator. The first element is provided.



[5]

- 3 A declarative programming language is used to represent the following knowledge base:

```

01 person(jane).
02 person(ahmed).
03 person(caroline).
04 person(stuart).
05 food(chocolate).
06 food(sushi).
07 food(pizza).
08 food(chilli).
09 likes(jane, pizza).
10 likes(ahmed, chocolate).
11 likes(ahmed, pizza).
12 likes(jane, chilli).
13 likes(stuart, sushi).
14 dislikes(stuart, chocolate).
15 dislikes(jane, sushi).
16 dislikes(caroline, pizza).

```

These clauses have the following meanings:

Clause	Explanation
01	Jane is a person
05	Chocolate is a food
09	Jane likes pizza
14	Stuart dislikes (does not like) chocolate

- (a) Mimi is a person who likes chocolate but does not like sushi or lettuce.

Write additional clauses to represent this information.

- 17
- 18
- 19
- 20
- 21

[5]

- (b) Using the variable PersonName, the goal:

likes(PersonName, pizza).

returns:

PersonName = jane, ahmed.

Write the result that is returned by the goal:

likes(ahmed, FoodItem).

FoodItem = [2]

- (c) B might like A, if B is a person, A is a food and B does not dislike A.

Write this as a rule.

might_like(.....,)

IF
.....
..... [6]

- 4 The following table shows part of the instruction set for a processor. The processor has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

- (a) A program stores a letter. The user is allowed nine attempts to guess the stored letter. The program outputs "?" and the user guesses a letter. If the user guesses the letter, the program outputs "*".

The following is pseudocode for this program.

```

REPEAT
    OUTPUT '?'
    INPUT GUESS
    IF GUESS = LETTERTOGUESS
        THEN
            OUTPUT '*'
            BREAK
    ELSE
        ATTEMPTS ← ATTEMPTS + 1
    ENDIF
UNTIL ATTEMPTS = 9

```

Write this program. Use the op codes from the instruction set provided.

Label	Op code	Operand	Comment
START:	LDM	#63	// load ASCII value for '?'
			// OUTPUT '?'
			// input GUESS
			// compare with stored letter
			// if correct guess, go to GUESSED
			// increment ATTEMPTS
			// is ATTEMPTS = 9 ?
			// if out of guesses, go to ENDP
			// go back to beginning of loop
GUESSED:	LDM	#42	// load ASCII for '*'
			// OUTPUT '*'
ENDP:	END		// end program
ATTEMPTS:		0	
LETTERTOGUESS:		'a'	

[11]

- (b) Five numbers are stored, starting in the location labelled NUMBERS. A program is needed to multiply each of the numbers by 4 and store them back in their original location.

Write this program. Use the op codes from the instruction set on the opposite page.

Label	Op code	Operand	Comment
START:			// initialise the Index Register
			// load the value from NUMBERS
			// multiply by 4
			// store the new value in NUMBERS
			// increment the Index Register
			// increment COUNT
			// is COUNT = 5 ?
			// repeat for next number
ENDP:	END		
COUNT:	0		
NUMBERS:	22		
	13		
	5		
	46		
	12		

[10]

Instruction		Explanation
Op code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

5 Large development projects require careful resource management.

- (a) (i) Name an appropriate project management tool that helps the manager to work out the estimated length of time it takes for the project to complete.

.....
.....

[1]

- (ii) Explain how, during the planning stage of the project, the manager would use the tool you named in part (a)(i).

.....
.....
.....
.....
.....

[3]

- (b) (i) Different programmers have been writing independent modules. The modules now need to be combined to create the final system.

Name the type of testing required at this stage.

.....
.....

[1]

- (ii) Name the final testing stage required before the system becomes operational.

.....
.....

[1]

- 6 A programmer wants to create a computer simulation of animals searching for food in a desert. The desert is represented by a 40 by 40 grid. Each position in the grid is represented by a pair of coordinates. 'A' represents an animal and 'F' represents food. At the start of the simulation, the grid contains 5 animals and 1 food source.

The following is an example of part of the grid.

	0	1	2	3	4	...	37	38	39
0	A					..			
1			F			..			
2						..		A	
3				A		..			
..
38				A		..	A		
39						..			

A timer is used. In each time interval, each animal randomly moves 0 or 1 position in a random direction. The program generates this movement by computing two random numbers, each of which can be -1, 0 or 1. The program adds the first random number to the across number and the second random number to the down number representing the animal's position.

For example:

- if 0 and 1 are generated, the across value does not change, the down value increases by 1
- if -1 and 1 are generated, the across value decreases by 1, and the down value increases by 1.

Each animal has an individual score. If the animal moves to a position in the grid with food ('F'):

- the animal's score increases by 1
- the food disappears
- one new animal ('A') is randomly generated and added to the grid (to a maximum of 20 animals)
- one new food ('F') is randomly generated and added to the grid.

The simulation is to be implemented using object-oriented programming.

The programmer has designed two classes, Desert and Animal.

The Desert class consists of:

- attributes
 - Grid
 - StepCounter
 - AnimalList
 - NumberOfAnimals
- methods
 - Constructor
 - IncrementStepCounter
 - GenerateFood
 - DisplayGrid

Each attribute consists of a value and a get and set method that allow access to the attributes.

The following table describes the attributes and methods for the Animal class.

Identifier	Data type	Description
Constructor()		<p>Instantiate an object of the Animal class</p> <ul style="list-style-type: none"> Generate a pair of random numbers between 0 and 39. Place animal at that random position. Initialise the animal's score to 0.
EatFood()		<ul style="list-style-type: none"> Delete the food. Increase the score of the animal that called the method. Call the GenerateFood method of the Desert class. Call the Constructor method of the Animal class.
Move()		<ul style="list-style-type: none"> Call the GenerateChangeInCoordinate method for each coordinate (across or down number) of the animal's position. Moves the animal to the new space. If there is food in the new position, call the EatFood method.
Score	INTEGER	Initialised to 0
Across	INTEGER	The across value, between 0 and 39
Down	INTEGER	The down value, between 0 and 39

- (a) Write **program code** to declare the attributes and constructor for the `Animal` class.

You only need to write the set and get methods for the attribute Across.

You should also write:

- the constructor for the class
 - set and get methods for the `Across` attribute only.

Programming language

Program code

.[6]

(b) The Constructor method of the Desert class:

- initialises an empty grid
 - creates 5 animal objects which are added to the `AnimalList` (an array of animal objects currently on the grid)
 - generates one food
 - sets the `StepCounter` to 0.

Write program code for the Constructor method.

Programming language

Program code

[5]

. [5]

(c) (i) The function GenerateChangeInCoordinate:

- receives a coordinate (across or down number) as a parameter
 - checks whether the coordinate's value is at a boundary of the grid
 - returns a random change (-1 , 0 or 1) that will keep the animal's position within the grid.

Write program code for the `GenerateChangeInCoordinate` function.

Programming language

Program code

[4]

.[4]

- (ii) The Move method uses the GenerateChangeInCoordinate function to calculate the new Across and Down values for an animal. If there is food in the new position in the grid, the animal eats the food.

Write program code for the Move method.

Programming language

Program code

[4]

. [4]

- (d) The programmer plans to add a graphic display to the program. The programmer will make use of a program library.

Explain what is meant by a program library.

[2]

.[2]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

CANDIDATE
NAME

--	--	--	--	--

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

May/June 2021

2 hours

You must answer on the question paper.

No additional materials are needed.

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.

This document has **20** pages. Any blank pages are indicated.

- 1 A vending machine allows users to insert coins to purchase an item.

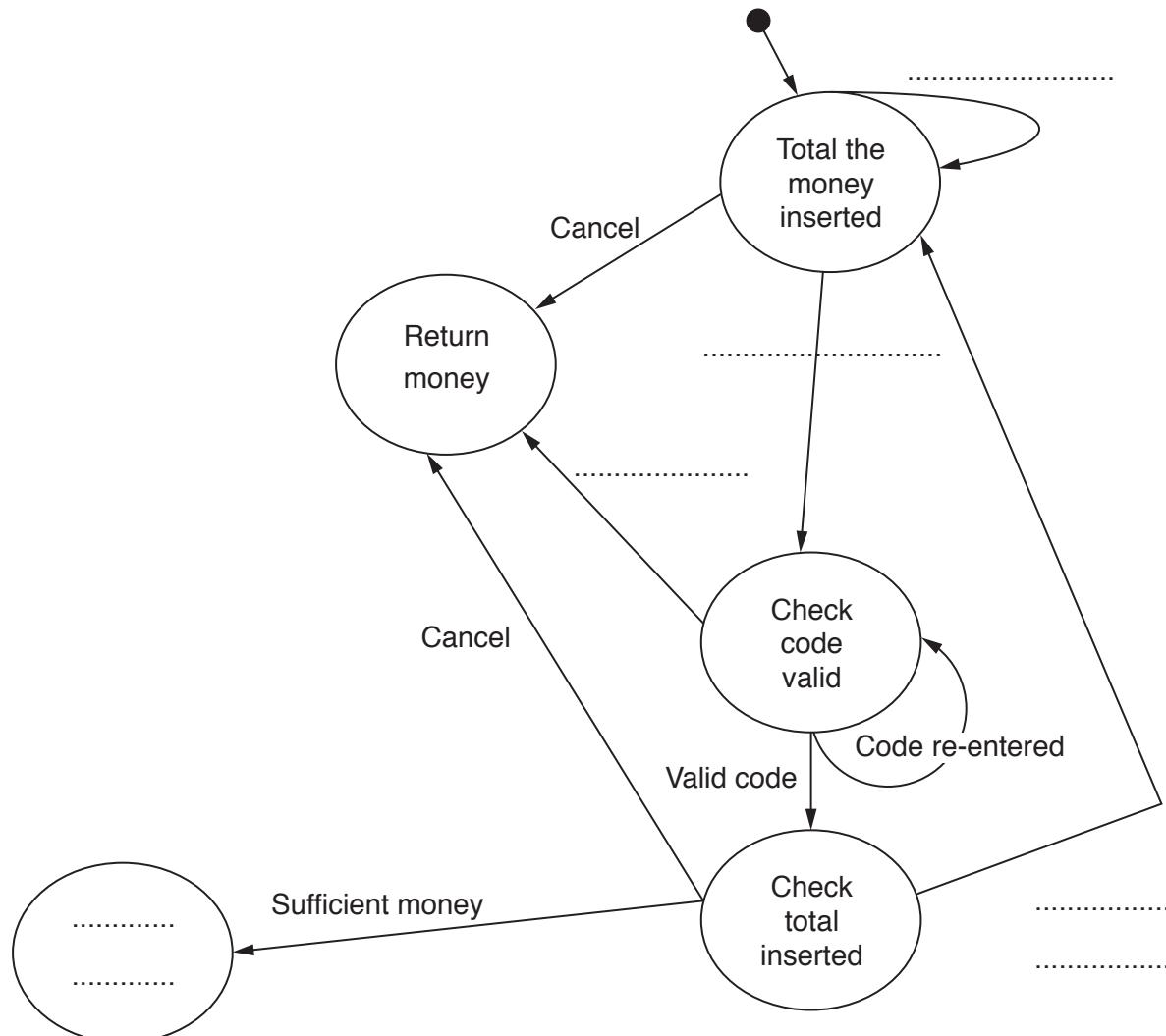
The user then enters the code for the item they would like the machine to dispense (give out). The user must re-enter the code until it is valid.

If the code is valid but the user has not inserted enough money for the item chosen, the machine waits for more coins to be inserted. The user then has to re-enter the code.

The user can press cancel at any time to return the money inserted into the machine.

- (a) The state-transition diagram shows the different states of the vending machine.

Complete the state-transition diagram.



[5]

- (b) The vending machine is part of a program that is written using object-oriented programming (OOP). The vending machine makes use of two classes that are described in the following tables.

All attributes are declared as private.

foodItem	
name : STRING	// the name of the item of food
code : STRING	// the code to be entered for that item to be // selected
cost : REAL	// the cost of the item
constructor(nameP, codeP, costP)	// creates an instance of foodItem // takes the name, code and cost as parameters
getCode() getCost() getName()	// returns the code for the item // returns the cost of the item // returns the name of the item

vendingMachine	
items : ARRAY[0:3] OF foodItem moneyIn : REAL	// stores four items of type foodItem // stores the total money inserted by the // user, initialised to 0 in the constructor
constructor(item1, item2, item3, item4)	// creates an instance of vendingMachine, // takes four objects of type foodItem as // parameters and stores them in array items
insertMoney()	// takes the value of the coin as a parameter // and adds it to moneyIn
checkValid ()	// takes a code as a parameter and checks it is // valid against the food item codes
getItemName()	// takes the array index as a parameter and // returns the name of the food items

- (i) Write **program code** to declare the class vendingMachine. You are only required to write program code for the attribute declarations and the constructor.

If you are writing in Python, include attribute declarations using comments.

Use your programming language's constructor method.

Programming language

Program code

[4]

[4]

- (ii) The method `checkValid()` takes the food item code as a parameter. It checks the code against each element in `items` and returns:

- -1 if the code is not valid
 - -2 if the code is valid, but the `moneyIn` is less than the cost of the item
 - the index of the item, if the code is valid and the `moneyIn` is greater than or equal to the cost of the item.

Write **program code** for the method `checkValid()`.

Programming language

Program code

[5]

- (iii) Four objects of type `foodItem` are declared with the identifiers:

chocolate, sweets, sandwich, apple

Write **program code** to declare an instance of vendingMachine with the identifier machineOne and the objects: chocolate, sweets, sandwich, apple.

Programming language

Program code

[2]

[2]

2 Peter uses a record structure, `customer`, to store data about customers. The data includes:

- a unique customer ID between 10 000 and 99 999
- the customer's first name
- the customer's last name
- the customer's telephone number (for example, +44 1234567891).

(a) Write **pseudocode** to define the record type `customer`.

.....

 [3]

(b) The customer records are stored in a random file. The location of each record is calculated as a hash value using:

$$(\text{customer}. \text{customerID} \bmod 1000) + 2$$

(i) Calculate the hash value for each of the customer IDs in the following table.

Customer ID	Hash value
40125	
10131	

[1]

(ii) Two or more records could have the same hash value that results in a collision.

Explain how the hashing algorithm can be designed to handle collisions.

.....

 [3]

(iii) The function, getCustomer():

- takes the customer ID as a parameter
 - passes the customer ID to the function `getRecordLocation()`, which returns the calculated hash value
 - reads and returns the record from the hashed location in the file `customerRecords.dat`

You can assume that both the file and the record being accessed exist.

Write pseudocode for the function `getCustomer()`.

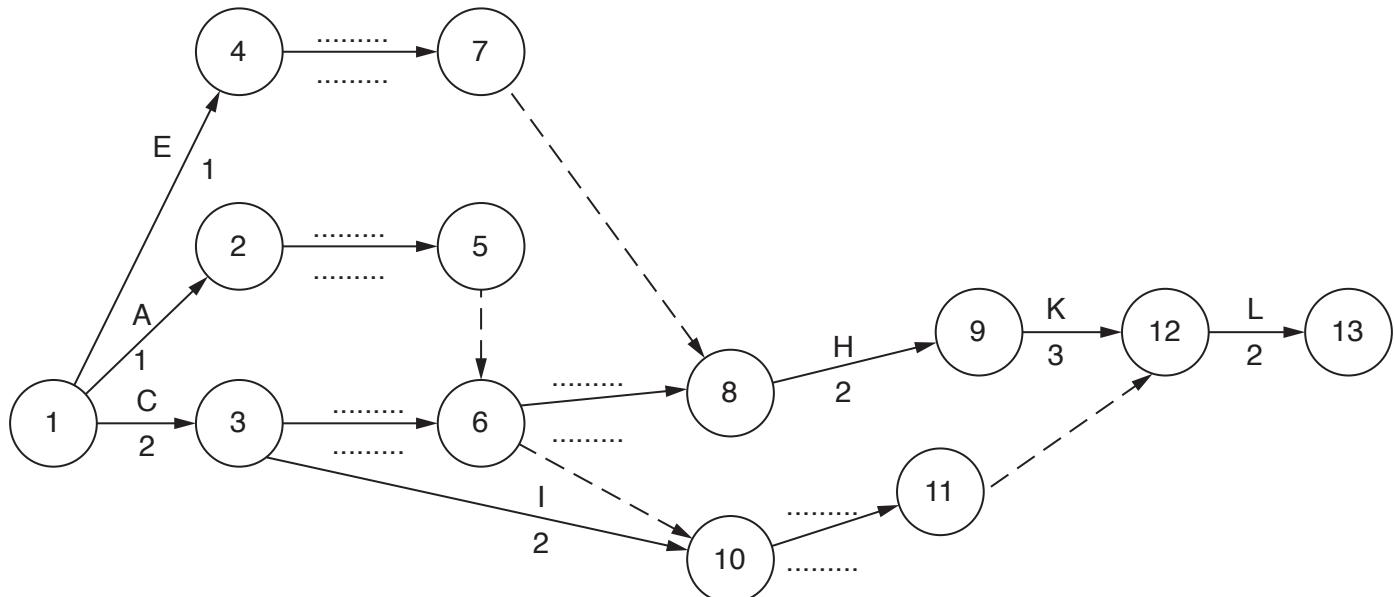
[5]

- 3 Alix manages a team of programmers who are creating a new computer game.

Alix has listed some of the tasks, along with their estimated time to complete and their immediate predecessors in the following table:

Task	Description	Predecessors	Time to complete (weeks)
A	Design character	–	1
B	Program character movement	A	1
C	Design level 1	–	2
D	Program level 1	C	2
E	Design robot	–	1
F	Program robot movement	E	1
G	Integrate character in level 1	B, D	2
H	Integrate robot in level 1	F, G	2
I	Design level 2	C	2
J	Program level 2	D, I	2
K	Test level 1	H	3
L	Integrate character and robot into level 2	J, K	2

- (a) Complete the Program Evaluation Review Technique (PERT) chart for the tasks in the table.



[5]

- (b) Explain how the tasks in the table can be divided between the team to allow concurrency of tasks.

.....
.....
.....
.....
.....
..... [2]

- (c) Explain the benefits of the team using program libraries in the development of the program.

.....
.....
.....
.....
.....
..... [3]

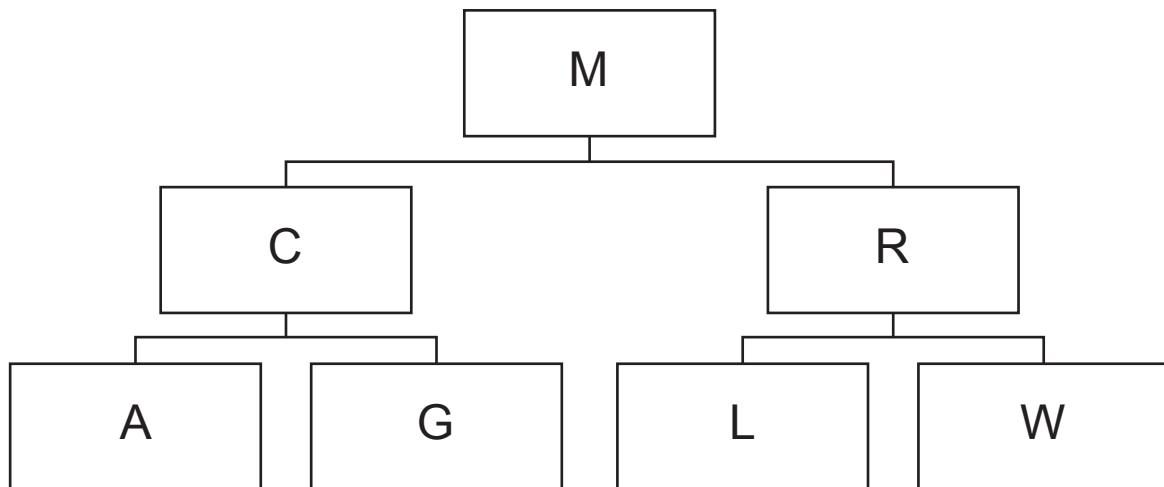
- (d) Identify **two** features in an editor that the developers can use to help them create their programs.

Feature 1

Feature 2

[2]

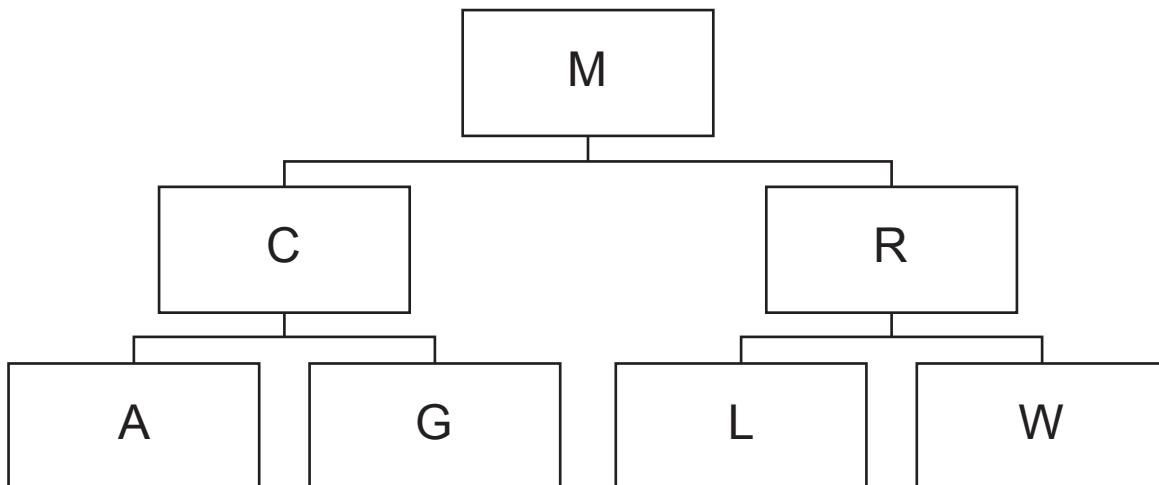
- 4 Chon creates a binary tree structure to store options that the user can select from a menu (M) in his program.



- (a) There are four new options that need to be added.

If option G is selected, the user must choose either option D or option H. If option L is selected, the user must choose either option J or option P.

Complete the following binary tree by adding options D, H, J and P.



- (b) Each node in the binary tree is stored using the following record structure:

```

TYPE node

    leftPointer : INTEGER
    data : STRING
    rightPointer : INTEGER

ENDTYPE

```

The tree is stored as a 1D array, `binaryTree`. Null pointers are represented by `-1`.

- (i) The table shows the contents of the three fields in each record stored in the 1D array `binaryTree`.

Complete the table to show the contents of `binaryTree` from part (a).

<code>rootPointer</code>	<code>freePointer</code>	Index	<code>leftPointer</code>	<code>data</code>	<code>rightPointer</code>
		0		M	
		1		C	
		2		A	
		3		L	
		4		G	
		5		R	
		6		W	
		7		J	
		8		D	
		9		P	
		10		H	
		11			

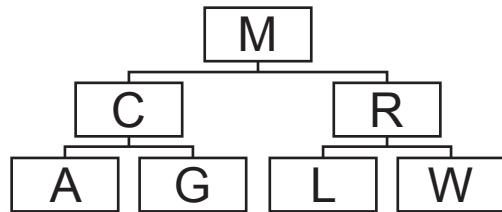
[4]

- (ii) Write **pseudocode** to declare the array `binaryTree` to store up to 100 objects of type `node`.

[2]

[2]

- (iii) A pre-order traversal on the following tree would output M C A G R L W



The pre-order traversal can be written as a recursive procedure:

1. output the root node
 2. follow the left pointer and repeat from step 1
 3. follow the right pointer and repeat from step 1.

Complete the **pseudocode** recursive procedure `preOrder()`.

```
PROCEDURE preOrder(BYVALUE rootPointer : INTEGER)
```

.....

.....
.....
.....
.....
.....
.....
.....
.....

ENDPROCEDURE

[6]

5 A binary search algorithm searches for data in a sorted array.

- (a) The pseudocode function `binarySearch()` performs a binary search to find a given value in the global array, `dataArray`. If the value is found, the function returns its index. If the value is not found, the function returns `-1`.

Complete the **pseudocode** for the function `binarySearch()`.

```

FUNCTION binarySearch(BYVALUE upper, lower, searchValue : INTEGER)
RETURNS INTEGER

DECLARE flag : INTEGER
DECLARE mid : INTEGER
flag ← -2
mid ← 0
WHILE flag <> -1
    mid ← lower + ((upper - lower) ..... )
    IF upper < lower
        THEN
            RETURN .....
    ELSE
        IF dataArray(mid) < searchValue
            THEN
                ..... ← .....
        ELSE
            IF dataArray(mid) > searchValue
                THEN
                    ..... ← .....
            ELSE
                RETURN .....
            ENDIF
        ENDIF
    ENDIF
ENDWHILE
ENDFUNCTION

```

[4]

- (b) The binary search algorithm can be written recursively.

Write **program code** for a recursive function `recursiveBinarySearch()`.

Programming language

Program code

[5]

- 6 The table shows assembly language instructions for a processor that has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

	Instruction		Explanation
Label	Op code	Operand	
	LDM	#n	Immediate addressing. Load the number n to ACC
	LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC
	LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC
	LDR	#n	Immediate addressing. Load the number n to IX
	STO	<address>	Store contents of ACC at the given address
	ADD	<address>	Add the contents of the given address to ACC
	INC	<register>	Add 1 to the contents of the register (ACC or IX)
	AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
	XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>
	OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>
	OUT		Output to screen the character whose ASCII value is stored in ACC
	CMP	<address>	Compare the contents of ACC with the contents of <address>
	CMP	#n	Compare the contents of ACC with number n
	JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
	JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
	JMP	<address>	Jump to the given address
	END		Return control to the operating system
<label>:	<op code>	<operand>	Labels an instruction
<label>:	<data>		Gives a symbolic address <label> to the memory location with contents <data>

An algorithm takes each letter of a stored 5-letter word and checks if the letter is upper case.

If the letter is upper case, it outputs the letter.

If the letter is not upper case, it converts the letter to upper case and then outputs it.

All ASCII upper case letters have 010 as the three most significant bits.

Assume each letter is alphabetic.

Complete the assembly language program for the algorithm described using the instruction set provided on the previous page.

Instruction			Comment	
Label	Op code	Operand		
	LDR	#0	// load zero to IX	
			// load count and check if it is 5	
	JPE	endP	// jump to end	
	LDX	word	// load letter from indexed address word	
			// check if it is upper case	
	CMP	#0		
	JPE	output	// jump to output if it is upper case	
	LDX	word	// load letter from indexed address word	
			// convert to upper case	
output:	OUT		// output the character	
			// increase count by 1	
	INC	IX	// increase IX by 1	
	JMP	start	// return to start	
endP:	end		// end the program	
word:	B01001000			
	B01101111			
	B01110101			
	B01110011			
	B01100101			
mask1:	B00100000			
mask2:	B11011111			
count:	0			

[6]

- 7 Giles is writing a program that uses a stack.

The stack stores up to 1000 integers in the 1D array, `stackArray`.

- (a) The procedure `setUpStack()` takes two parameters:

- the array, `stackArray`
- a pointer to the last element pushed onto the stack, `topOfStack`

The procedure initialises all array elements to -1 and the pointer to -1 .

Write **pseudocode** for the procedure `setUpStack()`.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [3]

- (b) The function `pop()` pops and returns the item from the top of the stack. If the stack is empty, it returns -1 .

Write **pseudocode** for the function `pop()`.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [3]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

May/June 2015

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **15** printed pages and **1** blank page.

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language

- 1 A petrol filling station has a single self-service petrol pump.

A customer can use the petrol pump when it is ready to dispense petrol.

The pump is in use when the customer takes the nozzle from a holster on the pump.

The pump dispenses petrol while the customer presses the trigger on the nozzle.

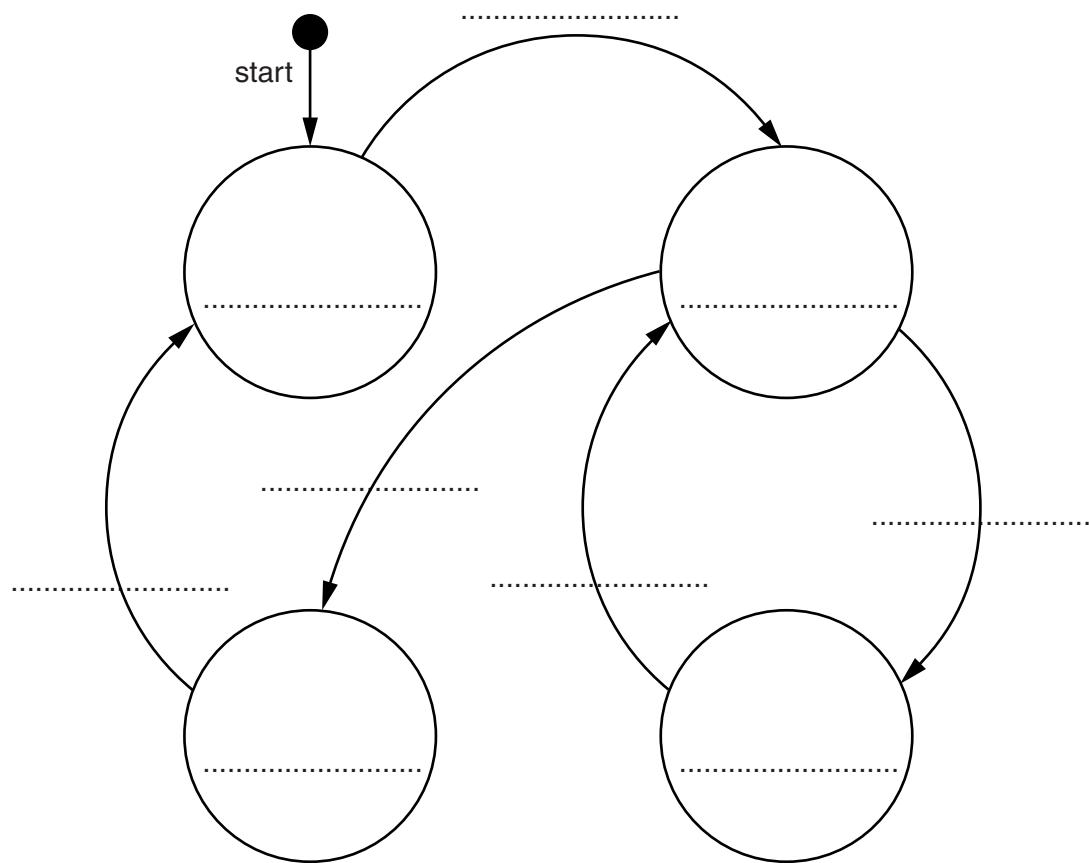
When the customer replaces the nozzle into the holster, the pump is out of use.

The cashier must press a reset button to make the pump ready for the next customer to use.

The petrol pump's four possible states and the transition from one state to another are as shown in the table below.

Current state	Event	Next state
Pump ready	Take nozzle	Pump in use
Pump in use	Press trigger	Pump dispensing
Pump dispensing	Stop pressing trigger	Pump in use
Pump in use	Replace nozzle	Pump out of use
Pump out of use	Reset pump display	Pump ready

Complete the state transition diagram for the petrol pump:



[9]

- 2** A declarative programming language is used to represent the knowledge base shown below:

```

01 dairy_product(cheese).
02 meat(beef).
03 meat(chicken).
04 meat(lamb).
05 made_with(burger, beef).
06 made_with(kofta, lamb).
07 made_with(quiche, cheese).
08 made_with(quiche, egg).
09 made_with(quiche, flour).

```

These clauses have the following meaning:

Clause	Explanation
01	Cheese is a dairy product
02	Beef is a meat
05	A burger is made with beef

- (a)** More facts are to be included.

Laasi is made with the dairy products milk and yogurt.

Write additional clauses to record this.

10

.....

11

.....

12

.....

13

..... [4]

- (b) Using the variable `TypeOfMeat`, the goal

`meat (TypeOfMeat)`

returns

`TypeOfMeat = beef, chicken, lamb`

Write the result returned by the goal:

`made_with (quiche, Ingredient)`

`Ingredient = [2]`

- (c) Complete the rule to list the dishes made with meat.

`contains_meat (Dish)`

`IF [4]`

.....

.....

.....

..... [4]

- 3 An insurance company calculates the cost of car insurance from a basic price.

The driver may:

- get a discount on the basic price of the insurance
- have to pay an extra charge

The decision is arrived at as follows:

- for a driver aged 25 or over:
 - 5% discount if no previous accident
 - no discount if a previous accident
- for a driver under the age of 25:
 - 5% discount if no previous accident and licence held for 3 or more years
 - no discount if a previous accident but licence held for 3 or more years
 - no discount if no previous accident but licence held for less than 3 years
 - 10% extra charge if a previous accident and licence held for less than 3 years

- (a) Complete the decision table.

Conditions	Age under 25	Y	Y	Y	Y	N	N	N	N
	Previous accident	Y	Y	N	N	Y	Y	N	N
	Licence held for 3 or more years	Y	N	Y	N	Y	N	Y	N
Actions	10% extra charge								
	No discount								
	5% discount								

[6]

- (b) Simplify your solution by removing redundancies.

Conditions	Age under 25								
	Previous accident								
	Licence held for 3 or more years								
Actions	10% extra charge								
	No discount								
	5% discount								

[3]

- (c) The simplified table produced in part (b) is used as a design for program code.

The following identifier table shows the parameters to be passed to the function CostPercentageChange. This function returns the percentage change from the basic price as an integer. A discount should be shown as a negative integer. An extra charge should be shown as a positive integer.

Identifier	Data type	Comment
DriverAge	INTEGER	Age of driver in years
HadAccident	BOOLEAN	Whether driver has had a previous accident
YearsLicenceHeld	INTEGER	Number of years the driver has held licence

Write **program code** for this function.

Programming language

[6]

[6]

- 4 A sports club stores data about its members. A program is to be written using an object-oriented programming language.

A Member class is designed. Two subclasses have been identified:

- FullMember
- JuniorMember

- (a) Draw an inheritance diagram for these classes.

[3]

- (b) The design for the Member class consists of

- properties
 - MemberName
 - MemberID
 - SubscriptionPaid
- methods
 - SetMemberName
 - SetMemberID
 - SetSubscriptionPaid

Write **program code** for the class definition of the superclass Member.

(c) Additionally a DateOfBirth property is required for the JuniorMember class.

(i) Write **program code** for the class definition for the subclass `JuniorMember`.

[3]

(ii) Write **program code** to create a new instance of JuniorMember. Use identifier NewMember with the following data:
name Ahmed with member ID 12347, born on 12/11/2001, who has paid his subscription.

name Ahmed with member ID 12347, born on 12/11/2001, who has paid his subscription.

[3]

5 A stack Abstract Data Type (ADT) has these associated operations:

- create stack
- add item to stack (push)
- remove item from stack (pop)

The stack ADT is to be implemented as a linked list of nodes.

Each node consists of data and a pointer to the next node.

- (a) There is one pointer: the top of stack pointer, which points to the last item added to the stack.
 Draw a diagram to show the final state of the stack after the following operations are carried out.

```
CreateStack
Push ("Ali")
Push ("Jack")
Pop
Push ("Ben")
Push ("Ahmed")
Pop
Push ("Jatinder")
```

Add appropriate labels to the diagram to show the final state of the stack. Use the space on the left as a workspace. Show your final answer in the node shapes on the right:



[3]

- (b) Using pseudocode, a record type, Node, is declared as follows:

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

The statement

```
DECLARE Stack : ARRAY[1:10] OF Node
```

reserves space for 10 nodes in array Stack.

- (i) The CreateStack operation links all nodes and initialises the TopOfStackPointer and FreePointer.

Complete the diagram to show the value of all pointers after CreateStack has been executed.

Stack		
	Name	Pointer
TopOfStackPointer		
<input type="text"/>		
FreePointer		
<input type="text"/>		
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		
[10]		

[4]

- (ii) The algorithm for adding a name to the stack is written, using pseudocode, as a procedure with the header

```
PROCEDURE Push (NewName)
```

Where NewName is the new name to be added to the stack. The procedure uses the variables as shown in the identifier table.

Identifier	Data type	Description
Stack	Array[1:10] OF Node	
NewName	STRING	Name to be added
FreePointer	INTEGER	Pointer to next free node in array
TopOfStackPointer	INTEGER	Pointer to first node in stack
TempPointer	INTEGER	Temporary store for copy of FreePointer

```
PROCEDURE Push(BYVALUE NewName : STRING)
// Report error if no free nodes remaining
IF FreePointer = 0
    THEN
        Report Error
    ELSE
        // new name placed in node at head of free list
        Stack[FreePointer].Name ← NewName
        // take a temporary copy and
        // then adjust free pointer
        TempPointer ← FreePointer
        FreePointer ← Stack[FreePointer].Pointer
        // link current node to previous top of stack
        Stack[TempPointer].Pointer ← TopOfStackPointer
        // adjust TopOfStackPointer to current node
        TopOfStackPointer ← TempPointer
    ENDIF
ENDPROCEDURE
```

Complete the **pseudocode** for the procedure Pop. Use the variables listed in the identifier table.

```
PROCEDURE Pop()
```

```
    // Report error if Stack is empty
```

.....
.....
.....
.....

```
    OUTPUT Stack [ ..... ] .Name
```

```
    // take a copy of the current top of stack pointer
```

.....

```
    // update the top of stack pointer
```

.....
.....
.....

```
    // link released node to free list
```

.....
.....
.....

```
ENDPROCEDURE
```

[5]

- 6 A recursively defined procedure X is defined below:

```

PROCEDURE X(BYVALUE n : INTEGER)
  IF (n = 0) OR (n = 1)
    THEN
      OUTPUT n
    ELSE
      CALL X(n DIV 2)
      OUTPUT (n MOD 2)
    ENDIF
ENDPROCEDURE

```

- (a) Explain what is meant by recursively defined.

.....
..... [1]

- (b) Explain how a stack is used during the execution of a recursive procedure.

.....
.....
.....
.....
..... [2]

- (c) Dry run the procedure X by completing the trace table for the procedure call:

CALL X(40)

Call number	n	(n = 0) OR (n = 1)	n DIV 2	n MOD 2
1	40	FALSE	20	
2				
3				
4				
5				
6				

OUTPUT [6]

- (d) State the process that is carried out by procedure x.

[1]

- (e)** Write **program code** for procedure x.

Programming language

.....

.....

.....

.....

.....

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

CANDIDATE
NAME

--	--	--	--	--

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

May/June 2020

2 hours

You must answer on the question paper.

No additional materials are needed.

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.

This document has **16** pages. Blank pages are indicated.

1 Amar is alerted to a run-time error when he runs a program.

(a) A run-time error occurs when Amar attempts to open a file that does not exist.

State **three other** reasons why a run-time error may occur.

1

.....

2

.....

3

.....

[3]

(b) A program should be written with exception handling routines to manage run-time errors.

A program reads data from the text file MyData.txt. The program needs to report an exception if it attempts to open the file and the file does not exist.

Write **program code** to handle and report this exception.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

[2]

- 2 A business is developing a program that stores each customer's username and password in a hash table.

The hash table will be implemented as a 1D array, CustomerLogIn, of the custom data type CustomerRecord.

The declaration for CustomerRecord is:

```
TYPE CustomerRecord  
    DECLARE Username : STRING  
    DECLARE Password : STRING  
ENDTYPE
```

The hash table will store a maximum of 3000 records. The key field will be Username.

- (a) The program declares the hash table and initialises the username and password of all the records to an empty string.

Write **pseudocode** to declare **and** initialise the hash table.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

[4]

(b) (i) A function, SearchHashTable() will search for a customer's record in the hash table.

The function Hash():

- takes a Username as a parameter
- performs the hashing algorithm
- returns the calculated index of the username within the hash table.

The function SearchHashTable():

- takes the username to search for as a parameter
- uses Hash() to calculate the first index of this username within the hash table
- returns either the index of the username if found, or -1 if not found.

Complete the **pseudocode** for the function SearchHashTable().

```

FUNCTION SearchHashTable(BYVALUE SearchUser : STRING) RETURNS .....

DECLARE Index : INTEGER
DECLARE Count : INTEGER
Index ← ..... (SearchUser)
Count ← 0
WHILE (CustomerLogIn[Index] ..... <> ..... )
    AND(CustomerLogIn[Index].Username <> "")
    AND(Count < 2999)
    Index ← Index + 1
    Count ← Count + 1
    IF Index > .....
        THEN
            Index ← 0
    ENDIF
ENDWHILE
IF CustomerLogIn[Index].Username = .....
    THEN
        RETURN Index
    ELSE
        RETURN .....
    ENDIF
ENDFUNCTION

```

[7]

- (ii) Explain the purpose of the variable `Count` in the function `SearchHashTable()`.

.....
.....
.....
.....

[2]

- 3 Recursive algorithms can be used when creating programs.

- (a) Describe what is meant by a **recursive algorithm**.

.....
.....
.....
.....
.....
.....
.....

[3]

- (b) A string that is a palindrome, reads the same forwards as it does backwards. For example, the name Anna is a palindrome.

The function `Substring(Variable, StartingCharacter, NumberOfCharacters)` returns one or more characters from a string. The first character is at position 0.

For example, the string "Happy" is stored in the variable `Word`.

- `Substring(Word, 1, 1)` would return the character "a".
- `Substring(Word, 2, 3)` would return the characters "ppy".

The function `Length()` returns the length of the string as an integer. For example, `Length(Word)` returns 5.

The following is a recursive function to find out whether a string is a palindrome. The function returns `True` if the parameter is a palindrome, and returns `False` if it is not a palindrome.

Complete the **pseudocode** for the recursive algorithm to indicate whether a string is a palindrome.

```

FUNCTION IsPalindrome(CheckWord : STRING) RETURNS BOOLEAN
    IF ..... <= 1
        THEN
            RETURN .....
    ENDIF
    IF Substring(CheckWord, 0, 1) <>
        Substring(CheckWord, ..... (CheckWord)-1, 1)
        THEN
            RETURN .....
        ELSE
            RETURN .....(Substring(CheckWord, 1,
                Length(CheckWord)-2))
        ENDIF
    ENDFUNCTION

```

[5]

- (c) The function `FindPower()` is a recursive function that calculates the result of a base number to the power of the exponent. For example, the result of $2^4 = 16$, as $2*2*2*2 = 16$. In this example, 2 is the base number and 4 is the exponent.

The base number and the exponent are passed as parameters.

Write **pseudocode** for the recursive function `FindPower()`. Assume both the base and the exponent are positive integers.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [5]

- 4 (a) A tennis club has a booking form to book lessons with an instructor.

Club members can book up to five lessons using the booking form.

The customer details section has the data:

- name
- address
- telephone number.

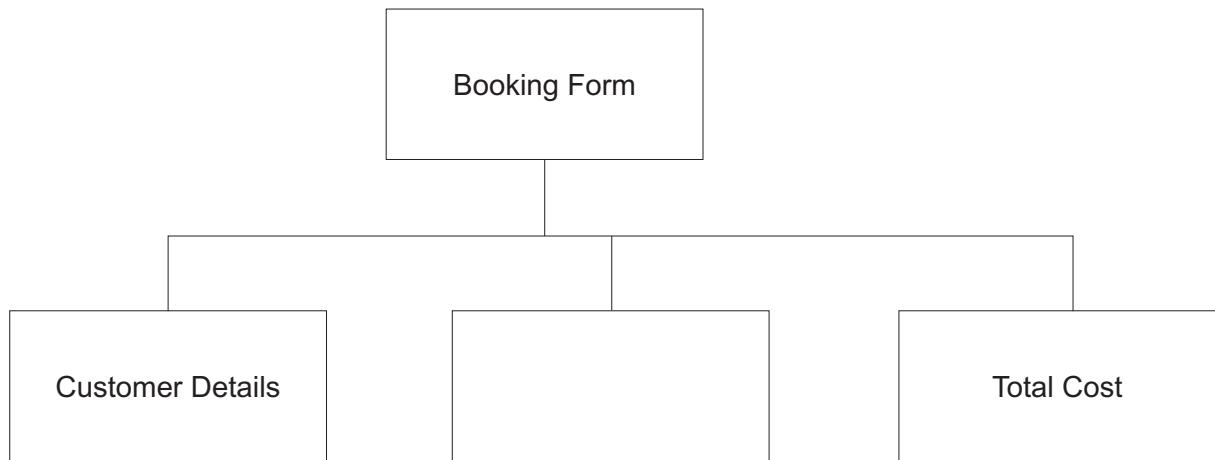
The lesson details section has the data:

- lesson type
- date and time
- lesson cost.

The cost of each lesson is dependent on the customer's type of membership. The membership can be bronze, silver or gold.

The total cost is also calculated.

Complete the following JSP data structure diagram for the booking form.



[7]

- (b) State **two** programming constructs that are shown in a JSP data structure diagram.

1

.....
2

.....
[2]

- 5 A declarative programming language is used to represent the following knowledge base.

```

01 person(william).
02 person(deeraj).
03 person(ingrid).
04 person(meghan).
05 country(england).
06 country(spain).
07 country(bangladesh).
08 country(new_zealand).
09 country(malaysia).
10 country(mauritius).
11 visited(william, spain).
12 visited(ingrid, new_zealand).
13 visited(deeraj, spain).
14 visited(meghan, spain).

```

These clauses have the following meanings:

Clause	Meaning
02	Deeraj is a person
05	England is a country
11	William has visited Spain

- (a) Gina is a person who has visited Cyprus.

Write additional clauses to represent this information.

15

16

17

[3]

- (b) Write the result returned by the goal:

visited(X, spain).

X = [2]

- (c) P might visit C, if P is a person, C is a country and P has not visited C.

Write this as a rule.

mightvisit(P , C)

IF

.....

..... [4]

- 6 Object-oriented programming has several features. These include containment, classes, methods and properties.

(a) Describe what is meant by **containment**.

.....
.....
.....
.....
..... [3]

(b) Identify **two other** features of object-oriented programming.

- 1
2 [2]

- 7 A programmer is creating a computer game. The programmer has designed the class, Character, for the characters in the game.

The following class diagram shows the design for the Character class.

Character	
Name : STRING	// initialised in constructor to the parameter value passed to the constructor
Skill : INTEGER	// initialised in constructor to 0
Health : INTEGER	// initialised in constructor to 50
Shield : INTEGER	// initialised in constructor to a random value between 1 and 25 (inclusive)
Constructor()	// method used to create and initialise an object
GetName()	// returns Name value
GetSkill()	// returns Skill value
GetHealth()	// returns Health value
GetShield()	// returns Shield value
SetSkill()	// increases Skill by the parameter value
SetHealth()	// increases or decreases Health by the parameter value
SetShield()	// increases or decreases Shield value by the parameter value

- (a) Write **program code** for the Constructor() method. Use the appropriate constructor method for your chosen programming language.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....
.....
.....
.....
..... [5]

- (b) Write **program code** for the `GetSkill()` method.

Programming language

Program code
.....
.....
.....
.....
..... [2]

- (c) The method `SetSkill()` validates the parameter value and updates the value of `Skill`.

The method is passed an `INTEGER` parameter that must be between 10 and 25 (inclusive). A value outside of this is not valid.

If the parameter value is valid, the method will increase `Skill` by the parameter value. The maximum value that `Skill` can be increased to is 200. For example:

- Skill currently stores 180
 - it is passed a valid parameter value of 25
 - Skill will now store 200.

The method must return:

- -1 if the parameter value is not valid
 - 1 if the value of Skill is updated **and** Skill is less than 200
 - 0 if the value of Skill is 200.

Write program code for the SetSkill() method.

Programming language

Program code

[6]

[6]

- (d) There are five characters in the game. All the character objects are stored in a 1D array.

Write **pseudocode** to declare the array, CharacterArray, to store the five character objects.

.....
.....

[2]

- (e) The game has the character with the name Victory.

Write **program code** to create the character Victory as an instance of the class Character. The object needs to be stored in the first element of the array CharacterArray.

Programming language

Program code

.....
.....
.....

[3]

Question 8 begins on the next page.

- 8 Files can be structured in serial, sequential or random format.

Tick () **one** box in each row to show whether the statement applies to **Serial**, **Sequential** or **Random** format.

Statement	Serial	Sequential	Random
Uses a hashing algorithm			
No key field is used when storing data, for example, it is stored in chronological order			
Collisions can occur			
Least efficient for a very large number of records			
Most efficient for a very large number of records			

[3]

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.



Cambridge International AS & A Level

CANDIDATE
NAME

--	--	--	--	--

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

May/June 2020

2 hours

You must answer on the question paper.

No additional materials are needed.

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.

This document has **16** pages. Blank pages are indicated.

- 1 Carlos is writing exception handling code for his program.

- (a) State what is meant by an **exception**.

..... [1]

- (b) Give **three** situations where an exception handling routine would be required.

1

2

3

[3]

- (c) Describe the benefits of using exception handling in a program.

.....
.....
..... [2]

- 2 (a) Programs can be written using recursion.

Tick (\checkmark) one or more boxes to show the features that **must** be included in a valid recursive algorithm.

Feature	Must be included
Incrementation	
General case	
Base case	
Selection case	
It calls itself	

[2]

- (b) The following recursive procedure outputs every even number from the positive parameter value down to and including 2.

The procedure checks if the integer parameter is an even or an odd number. If the number is odd, the procedure converts it to an even number by subtracting 1 from it.

The function MOD(ThisNum : INTEGER, ThisDiv : INTEGER) returns the remainder value when ThisNum is divided by ThisDiv.

Complete the **pseudocode** for the recursive procedure.

```

PROCEDURE Count (BYVALUE ..... : INTEGER)

    IF ..... (Number, 2) <> 0
        THEN
            Number ← Number - 1
        ENDIF
        OUTPUT .....
        IF Number > 0
            THEN
                ..... ( ..... - 1)
            ENDIF
    ENDPROCEDURE

```

[5]

- (c) A program allows guests to input a meal option at a wedding.

Guests can choose meal option 1 or meal option 2.

The program will keep count of the numbers of each meal option chosen.

The program ends when a value other than 1 or 2 is entered. It then outputs the count of each meal option.

```

PROCEDURE MealsCount (BYREF MealOption1 : INTEGER, MealOption2 : INTEGER)

DECLARE MealOption : INTEGER

DECLARE MoreMeals : BOOLEAN

MoreMeals ← True

WHILE MoreMeals = True

    INPUT MealOption

    IF MealOption = 1

        THEN

            MealOption1 ← MealOption1 + 1

    ELSE

        IF MealOption = 2

            THEN

                MealOption2 ← MealOption2 + 1

            ELSE

                OUTPUT MealOption1, " ", MealOption2

                MoreMeals ← False

            ENDIF

    ENDIF

ENDWHILE

ENDPROCEDURE

```

The program contains a conditional loop.

Use **pseudocode** to rewrite the conditional loop as a recursive algorithm.

```
PROCEDURE MealsCount (BYREF MealOption1 : INTEGER, MealOption2 : INTEGER)
```

```
    DECLARE MealOption : INTEGER
```

```
    .....  
    .....
```

```
    ENDPROCEDURE
```

[5]

- 3 A declarative programming language is used to represent the following knowledge base.

```

01 person(jessica).
02 person(pradeep).
03 person(steffi).
04 person(johann).
05 sport(football).
06 sport(hockey).
07 sport(cricket).
08 sport(volleyball).
09 plays(johann, football).
10 plays(steffi, cricket).
11 plays(jessica, football).
12 will_not_play(pradeep, cricket).

```

These clauses have the following meanings:

Clause	Meaning
01	Jessica is a person
05	Football is a sport
09	Johann plays football
12	Pradeep refuses to play cricket

- (a) Elle is a person who plays rugby but refuses to play hockey.

Write additional clauses to represent this information.

- 13
- 14
- 15
- 16

[4]

- (b) Write the result returned by the goal:

plays(X, football).

X = [1]

- (c) Y might play X, if Y is a person, X is a sport and Y does not refuse to play X.

Write this as a rule.

mightplay(Y , X)

IF

.....
..... [5]

- 4 Object-oriented programming has several features. These include inheritance, classes, methods and properties.

- (a) Describe what is meant by **inheritance**.

.....
.....
.....
..... [2]

- (b) Identify **two other** features of object-oriented programming.

1

2

[2]

- 5 A tennis club is developing a program to store details of the lessons it offers. The programmer has designed the class Lesson for the details of the lessons.

The following class diagram shows the design for the Lesson class.

Lesson	
LessonType : STRING	// initialised in constructor to the parameter // value passed to the constructor
Instructor : STRING	// initialised in constructor to the parameter // value passed to the constructor
Constructor()	// method used to create and initialise an // object
GetLessonType()	// returns LessonType value
GetInstructor()	// returns Instructor value
GetFee()	// returns the cost of a lesson
SetLessonType()	// sets the LessonType to the parameter value
SetInstructor()	// sets the Instructor to the parameter value

- (a) Write **program code** for the Constructor() method.

Use the appropriate constructor method for your chosen programming language.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

[3]

- (b) Write **program code** for the `GetLessonType()` method.

Programming language

Program code

.....
.....
.....
.....
.....
.....

[2]

- (c) Fee is the cost that a customer will pay for a lesson.

The method `GetFee()` validates the parameter value. The method is sent a parameter value that represents the skill level of the customer: beginner, intermediate or advanced.

The parameter will be a character:

- 'B' for beginner
 - 'I' for intermediate
 - 'A' for advanced.

The method must check the parameter value is a valid character ('B', 'I' or 'A') and return the correct fee. It must return -1 if it is not a valid character.

The fees are:

- \$45 for a beginner
 - \$50 for an intermediate
 - \$55 for an advanced.

Write program code for the GetFee () method.

Programming language

Program code

[5]

- (d) The tennis club only offers nine different types of lesson. The lesson objects are stored in a 1D array.

Write **pseudocode** to declare an array `LessonArray` to store the nine lesson objects.

.....
..... [2]

- (e) The tennis club has the lesson ‘Improve Your Serve’ that has David as the instructor.

Write **program code** to create the lesson ‘Improve Your Serve’ as an instance of the class `Lesson`. The object needs to be stored in the third element of the array `LessonArray`.

Programming language

Program code

.....
.....
..... [3]

- 6 A theatre company stores customer login details to allow customers to book tickets online.

A hash table stores login details for 2000 customers.

Each customer's details are stored in a record.

The declaration for CustomerRecord is:

```
TYPE CustomerRecord
  DECLARE UserID : STRING
  DECLARE PINNumber : INTEGER
ENDTYPE
```

A 1D array, CustomerDetails, is used to implement the hash table. CustomerDetails is a global array. The 1D array has 6000 elements.

- (a) The procedure InitialiseHashTable() initialises the hash table. UserID is initialised as an empty string, and PINNumber initialised to 0 for all of the records.

Write **pseudocode** for the procedure InitialiseHashTable().

.....

 [4]

- (b) The function InsertRecord() is used to insert a new record into the hash table.

The function, Hash():

- takes a UserID as a parameter
- performs the hashing algorithm
- returns the calculated index of the user ID within the hash table.

If the hash table is full, the function InsertRecord() returns -1. If there is space available in the hash table, the record is inserted, and it returns the position of this record in the array.

Complete the **pseudocode** for the function.

```

FUNCTION InsertRecord(NewRecord : CustomerRecord) RETURNS INTEGER

DECLARE Count : INTEGER
DECLARE Index : INTEGER
Count ← 0
Index ← Hash(.....)
WHILE (CustomerDetails[Index].UserID <> "") ..... (Count <= 5999)

    Index ← Index + 1
    Count ← Count + 1
    IF Index > 5999
        THEN
            .....
    ENDIF
ENDWHILE
IF Count > 5999
    THEN
        .....
ELSE
    CustomerDetails[.....] ← .....
    .....
ENDIF
ENDFUNCTION

```

[7]

- 7 (a) A shirt design company has an order form to order shirts. Customers can order multiple shirts using the same form.

The customer details section has the data:

- name
- address
- telephone number.

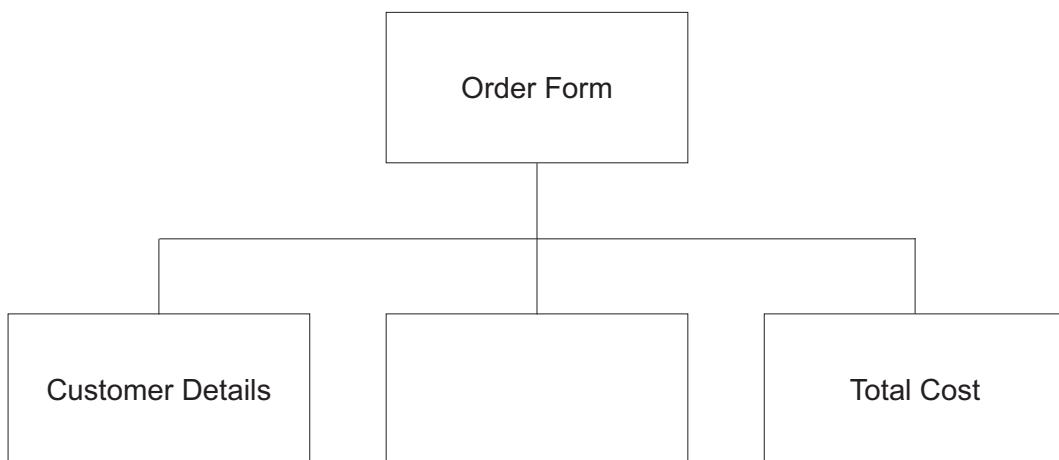
The order details section has the data:

- shirt ID
- colour
- cost.

A total cost for the order is also calculated.

The cost of each shirt is dependent on the size ordered. The sizes customers can order are small, medium and large.

Complete the following JSP data structure diagram for the order form.



[7]

- (b) Each customer's order is stored as a record in a file. The customers' orders are stored in the order in which they arrive in the file and no key field is used.

(i) Identify this type of file structure.

..... [1]

(ii) Identify **two other** types of file structure.

1

2

[2]

- (c) The procedure `UpdateTelephone()` allows the shirt company to update the record for a customer's details. The procedure will update the telephone number.

The program stores customer details as a custom data type, `Customer`.

The definition for this data type is:

```
TYPE Customer
```

```
    Name : STRING
```

```
    Address : STRING
```

```
    TelephoneNumber : STRING
```

```
ENDTYPE
```

The procedure `UpdateTelephone()` takes the customer record to be updated and the new telephone number as parameters. It then updates the telephone number in the record.

Complete the **pseudocode** for the procedure `UpdateTelephone()`.

```
PROCEDURE UpdateTelephone (..... ThisCustomer : Customer,  
..... NewTelephoneNumber : STRING)
```

```
.....
```

[3]

(d) The shirt company is looking to implement a system to reward customers. The system includes:

- 10% discount on orders over \$50
- free gift if order over \$50 and if the order is placed on a Monday
- additional 5% discount if a customer has a loyalty card
- free delivery for a customer with a loyalty card and spends over \$50.

Complete the following decision table for this system.

Conditions	Order over \$50	Y	Y	Y	Y	N	N	N	N
	Monday	Y	Y	N	N	Y	Y	N	N
	Loyalty card	Y	N	Y	N	Y	N	Y	N
	Additional 5% discount								
	10% discount								
	Free gift								
	Free delivery								

[4]

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

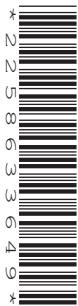
CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

October/November 2018

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

- 1 A declarative language is used to represent the following facts and rules about animals.

```

01 feature(dog, drinks_milk).

02 feature(dog, has_lungs).

03 feature(horse, has_lungs).

04 feature(tuna, lives_in_water).

05 feature(tuna, has_gills).

06 feature(crab, lives_in_water).

07 mammal(drinks_milk).

08 mammal(has_lungs).

09 fish(lives_in_water).

10 fish(has_gills).

11 is_a_mammal(X) IF (feature(X, Y) AND mammal(Y)) AND (feature(X, Z)
AND mammal(Z)).

```

These clauses are explained in the following table.

Clause	Explanation
01	A dog has the feature, drinks milk
07	A mammal drinks milk
11	X is a mammal, if: • X has the feature Y and a mammal has a feature Y, and • X has the feature Z and a mammal has the feature Z

- (a) More facts are to be included.

- (i) A bird has wings, and a bird lays eggs.

Write the additional clauses to record these facts.

12

13

[2]

- (ii) An eagle has all the features of a bird.

Write the additional clauses to record this fact.

14

15

[2]

- (b) (i) Using the variable B, the goal

```
feature(B, drinks_milk)
```

returns

B = dog

Write the result returned by the goal

```
feature(B, lives_in_water)
```

B = [2]

- (ii) Write a goal, using the variable C, to find the feature(s) of tuna.

..... [2]

- (c) An animal is a bird if it lays eggs **and** it has wings.

Complete the following rule.

is_a_bird(X) IF

..... [3]

- (d) Declarative programming and object-oriented programming are two examples of programming paradigms.

- (i) Define the term **programming paradigm**.

.....

..... [1]

- (ii) Give **two** examples of programming paradigms, other than declarative and object-oriented programming.

1

2

[2]

- 2 Kendra collects books. She is writing a program to store and analyse information about her books.

Her program stores information about each book as a record. The following table shows the information that will be stored about each book.

Field name	Description
Title	The title of the book
Author	The first listed author of the book
ISBN	A 13-digit code that uniquely identifies the book, for example: "0081107546738"
Fiction	If the book is fiction (TRUE) or non-fiction (FALSE)
LastRead	The date when Kendra last read the book

- (a) Write **pseudocode** to declare an Abstract Data Type (ADT) named Book, to store the information in the table.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[4]

- (b)** The records are stored in a random access file.

The function, `Hash()`, takes as a parameter the ISBN and returns the hash value.

The disk address of the record in the hash table is calculated as: ISBN modulus 2000 plus 1.

Write program code for the function Hash () .

Programming language

Program code

[4]

[4]

- (c) The random access file, MyBooks.dat, stores the data about the books in the format:

```
<Title>
<Author>
<ISBN>
<Fiction>
<LastRead>
```

A procedure, FindBook():

- prompts the user to input the ISBN of a book until the ISBN contains 13 numeric digits
 - uses the function `Hash()` to calculate the disk address of the record
 - reads the record for that book from `MyBooks.dat` into a variable of type `Book`
 - outputs all the data about the book.

Use **pseudocode** to write the procedure `FindBook()`.

You can assume that the record exists at the disk address generated.

[8]

. [8]

- 3 Joseph is taking a toy apart. Each time he removes an item from the toy, he writes the name of the item at the bottom of a paper list. When he rebuilds the toy, he puts the items back together working from the bottom of the list.

Joseph writes a computer program to create the list using a stack, `Parts`.

- (a) Describe a stack structure.

.....
..... [1]

- (b) The stack is represented as an array in the program, the first element in the array is [0].

The current contents of the stack, `Parts`, and its pointer, `StackPointer` are shown.

<code>StackPointer</code>	<input type="text" value="5"/>	<code>StackContents</code>
0		"Screw 1"
1		"Screw 2"
2		"Back case"
3		"Screw 3"
4		"Engine outer"
5		
6		
7		

- (i) Describe the purpose of the variable `StackPointer`.

.....
..... [1]

- (ii) The procedure `POP()` removes an item from the stack. The procedure `PUSH(<identifier>)` adds an item to the stack.

The current contents of the stack, `Parts`, and its pointer, `StackPointer` are shown.

StackPointer	<input type="text" value="5"/>	StackContents
0		"Screw 1"
1		"Screw 2"
2		"Back case"
3		"Screw 3"
4		"Engine outer"
5		
6		
7		

Use the table below to show the contents of the stack, `Parts`, and its pointer after the following code is run.

```

POP()
POP()
PUSH("Light 1")
PUSH("Light 2")
PUSH("Wheel 1")
POP()
POP()

```

StackPointer	<input type="text"/>	StackContents
0		
1		
2		
3		
4		
5		
6		
7		

- (c) A 1D array, Parts, is used to implement the stack. Parts is declared as:

```
DECLARE Parts : ARRAY[0 : 19] OF STRING
```

- (i) The procedure POP outputs the last element that has been pushed onto the stack and replaces it with a '*'.

Complete the **pseudocode** for the procedure POP.

```
PROCEDURE POP
```

```
IF ..... = .....
```

```
THEN
```

```
OUTPUT "The stack is empty"
```

```
ELSE
```

```
StackPointer ← .....
```

```
OUTPUT .....
```

```
Parts[StackPointer] ← .....
```

```
ENDIF
```

```
ENDPROCEDURE
```

[5]

- (ii) The procedure PUSH() puts the parameter onto the stack.

Complete the **pseudocode** for the procedure PUSH().

```
PROCEDURE PUSH(BYVALUE Value : String)
```

```
IF StackPointer > .....
```

```
THEN
```

```
OUTPUT "Stack full"
```

```
ELSE
```

```
..... ← .....
```

```
StackPointer ← .....
```

```
ENDIF
```

```
ENDPROCEDURE
```

[4]

- 4 The recursive algorithm for the `Calculate()` function is defined as follows:

```
01 FUNCTION Calculate(BYVALUE Number : INTEGER) RETURNS INTEGER  
02     IF Number = 0  
03         THEN  
04             Calculate ← -10  
05         ELSE  
06             Calculate ← Number * Calculate(Number - 1)  
07     ENDIF  
08 ENDFUNCTION
```

- (a) (i) State what is meant by a **recursive algorithm**.

..... [1]

- (ii) State the line number in `Calculate()` where the recursive call takes place.

..... [1]

Question 4(b) begins on the next page.

- (b) The function is called with Calculate(3).

Dry run the function **and** complete the trace table below. State the final value returned. Show your working.

```

01 FUNCTION Calculate(BYVALUE Number : INTEGER) RETURNS INTEGER
02     IF Number = 0
03         THEN
04             Calculate ← -10
05         ELSE
06             Calculate ← Number * Calculate(Number - 1)
07     ENDIF
08 ENDFUNCTION

```

Working

.....

.....

.....

Trace table:

Call number	Function call	Number = 0 ?	Return value

Final return value

[6]

- (c) A recursive algorithm within a subroutine can be replaced with an iterative algorithm.

(i) Describe **one** problem that can occur when running a subroutine that has a recursive algorithm.

[2]

[2]

- (ii) Rewrite the Calculate() function in **pseudocode**, using an **iterative algorithm**.

[5]

- 5 A game uses a set of cards. Each card has a number (between 0 and 9 inclusive) and a shape ("square", "triangle" or "circle").

The game is written using object-oriented programming.

The class, Cards, has the private properties:

- Number
- Shape

and the methods:

- Constructor()
- GetNumber()
- GetShape()

The purpose of each method in the class Cards is given in the following table.

Method	Purpose
Constructor()	Takes a number and a shape as parameters Checks that the number and the shape are valid and: <ul style="list-style-type: none"> • either assigns the parameters to Number and Shape • or reports an error.
GetNumber()	A public method that returns the number for that card.
GetShape()	A public method that returns the shape for that card.

- (a) Explain why the properties are private.

.....

[2]

- (b)** Write **program code** for the `Constructor()` method.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

[5]

- (c) Write program code for the GetNumber() method.**

Programming language

Program code

[2]

[2]

- (d) A card, OneS, has the value 1 for Number and the value "square" for Shape.

Write program code to instantiate an instance of Cards for Ones.

Programming language

Program code

[2]

[2]

- (e) The game has a function, `Compare()` that takes two cards as parameters and compares them.

If the cards are identical, the function outputs "SNAP" and returns -1. If they are not identical, and the card numbers are different, it returns the Number of the card with the higher value or the Number for the cards if they are the same.

Write program code for the Compare () function.

Programming language

Program code

[6]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

May/June 2018

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

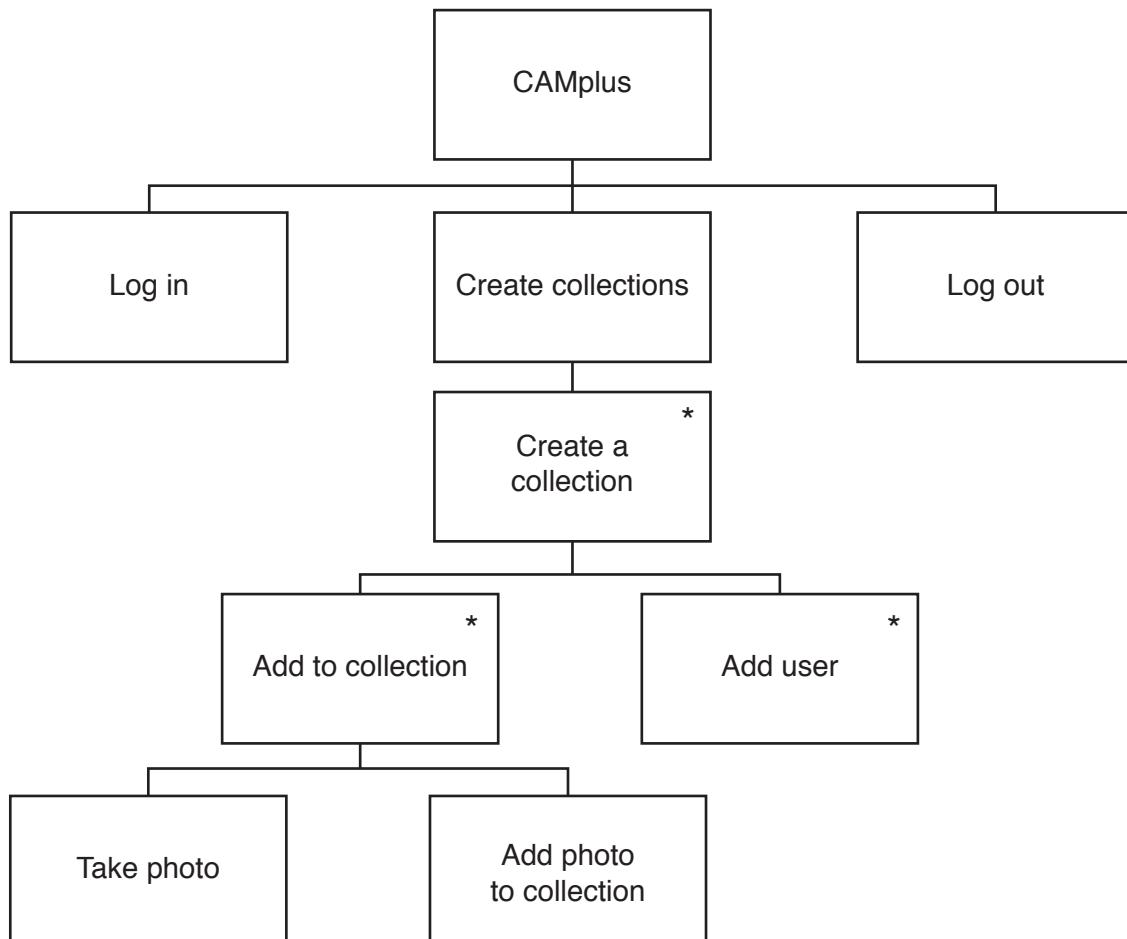
The maximum number of marks is 75.

This document consists of **22** printed pages and **2** blank pages.

- 1 Paul is using an application (app) called CAMplus. The app allows users to:

- log in
- create a new collection of photographs
- use the camera to take new photographs
- automatically add new photographs to the new collection
- share the new collection with other users
- start another collection or log out of the app.

The following JSP structure diagram represents the operation of CAMplus.



- (a) An algorithm has been written in pseudocode to represent the **Create collections** operation from the JSP structure diagram. The algorithm is incomplete.

Write **pseudocode** to complete this algorithm.

REPEAT

REPEAT

CALL TakePhoto

.....
OUTPUT "Do you want to take another photo?"

INPUT AddPhoto

UNTIL AddPhoto = "No"

REPEAT

.....
OUTPUT "Do you want to add another user?"

INPUT NewUser

UNTIL = "No"

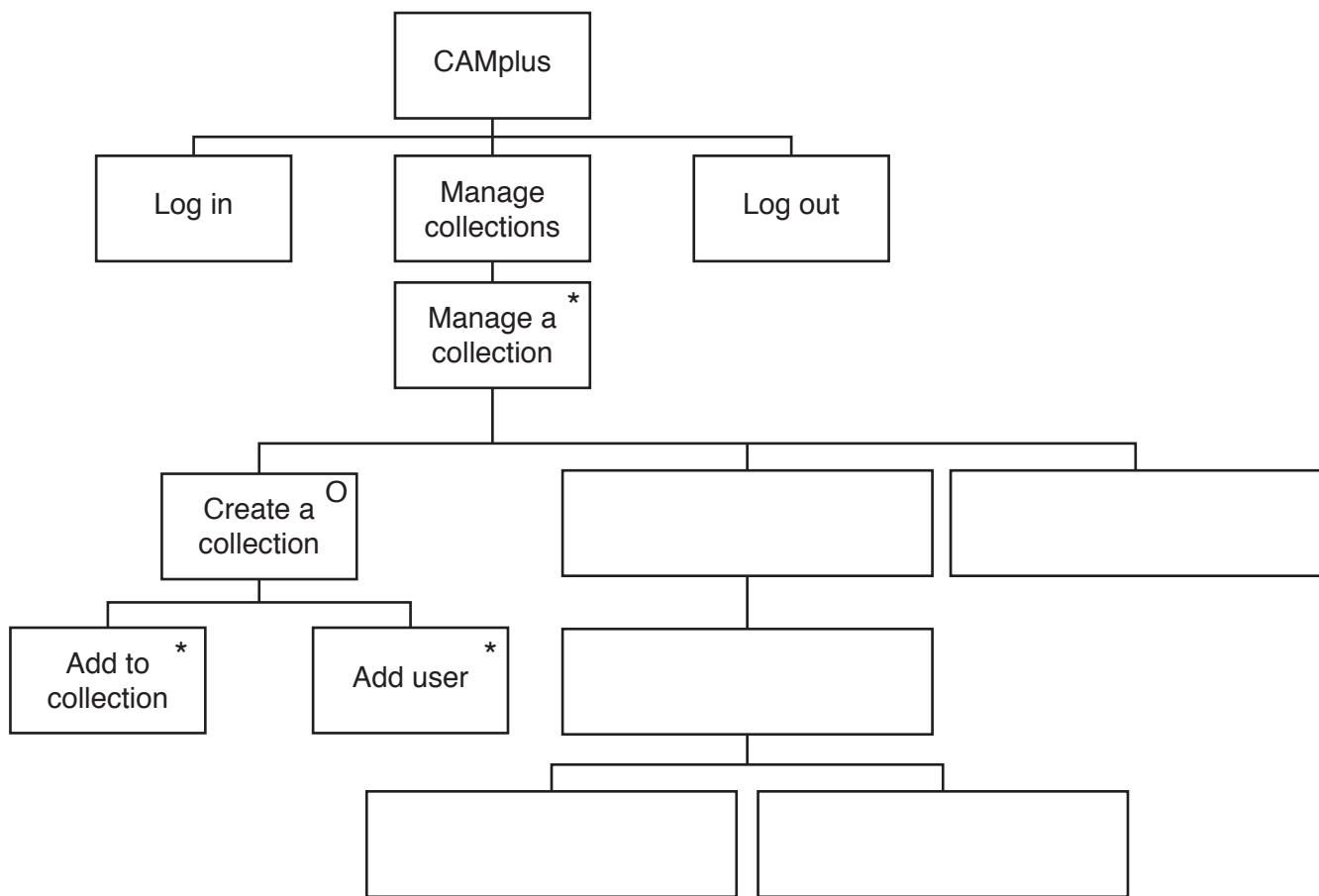
.....
OUTPUT "Do you want to create another collection?"

.....
UNTIL NewCollection = "No"

[4]

- (b) The app is updated. Paul can now add and delete photos from chosen collections. Paul can also delete collections.

Complete the JSP structure diagram to show the changes.



[5]

Question 2 begins on the next page.

- 2 A declarative language is used to represent the following facts and rules about iguanas and lizards.

```

01 has(reptile, cold_blood).
02 has(reptile, air_breathing).
03 has(reptile, scales).
04
05 is_a(squamata, reptile).
06 is_a(iguana, squamata).
07 is_a(lizard, squamata).
08 is_a(green_iguana, iguana).
09 is_a(cayman, iguana).
10 is_a(smooth_iguana, iguana).
11
12 maxsize(green_iguana, 152).
13 maxsize(cayman, 90).
14 maxsize(smooth_iguana, 70).

```

These clauses have the following meaning:

Clause	Explanation
01	A reptile has cold blood.
09	A cayman is a type of iguana.
12	The maximum size of a green iguana is 152 cm.

- (a) More facts are to be included.

A gecko is a type of lizard. It has a maximum size of 182 cm.

Write the additional clauses to record these facts.

15

16

[2]

- (b)** Using the variable R , the goal

`is_a(R, squamata).`

returns

`R = iguana, lizard`

Write the result returned by the goal

`is_a(T, iguana).`

`T = [2]`

- (c)** Write the goal, using the variable X , to find what a squamata is.

..... [2]

- (d)** All iguanas and lizards are squamata. All squamata are reptiles.

Write a recursive rule to make all lizards and iguanas inherit the properties of reptiles.

`has(X, Y)`

IF

.....
..... [3]

- (e)** State what the following goal returns.

`NOT(maxsize(cayman, 70)).`

..... [1]

- 3 The arrays PollData[1:10] and CardData[1:10] store data.

PollData

12	85	52	57	25	11	33	59	56	91
----	----	----	----	----	----	----	----	----	----

CardData

11	12	25	33	52	56	57	59	91	85
----	----	----	----	----	----	----	----	----	----

An **insertion sort** sorts these data.

- (a) State why it will take less time to complete an insertion sort on CardData than on PollData.

.....
..... [1]

- (b) The following pseudocode algorithm performs an insertion sort on the CardData array.

Complete the following **pseudocode** algorithm.

```

01  ArraySize ← 10
02  FOR Pointer ← 2 TO .....
03      ValueToInsert ← CardData[Pointer]
04      HolePosition ← .....
05      WHILE (HolePosition > 1 AND ( ..... > ..... ))
06          CardData[HolePosition] ← CardData[ ..... ]
07          HolePosition ← .....
08      ENDWHILE
09      CardData[HolePosition] ← .....
10  ENDFOR

```

[7]

- (c) (i) A binary search algorithm is used to find a specific value in an array.

Explain why an array needs to be sorted before a binary search algorithm can be used.

. [2]

- (ii) The current contents of CardData are shown.

11	12	25	33	52	56	57	59	85	91
----	----	----	----	----	----	----	----	----	----

Explain how a binary search will find the value 25 in CardData.

[4]

- (d) Complete this procedure to carry out a binary search on the array shown in part (c)(ii).

```

PROCEDURE BinarySearch(CardData, SearchValue)

DECLARE Midpoint : INTEGER

First ← 1

Last ← ARRAYLENGTH(.....)

Found ← FALSE

WHILE (First ≤ Last) AND NOT(Found)

    Midpoint ← .....

    IF CardData[Midpoint] = SearchValue

        THEN

            Found ← TRUE

        ELSE

            IF SearchValue < CardData[Midpoint]

                THEN

                    Last ← ......

                ELSE

                    First ← ......

                ENDIF

            ENDIF

        ENDIF

    ENDWHILE

ENDPROCEDURE

```

[4]

Question 4 begins on the next page.

- 4 X-Games is an international extreme sports competition.

A program will store and process data about the teams in the competition.

- Each team is made up of members.
- Members can be added and removed from each team.
- Each member has a first name, last name, date of birth and gender.
- Each member can be an official or a competitor.
- Each official has a job title and may be first-aid trained.
- Each competitor takes part in one sport.

The program is written using object-oriented programming.

The program can output the full name and date of birth of any member. For example, “Nadia Abad 16/05/1995”

An introduction about a team member can be output using their name. For example, “Hello, I’m Nadia Abad”.

The program outputs a different version of the introduction for a competitor. This version includes the competitor’s sport. For example, “Hello, I’m Sally Jones and my sport is Skateboard Park.”

- (a) Complete the following class diagram to show the attributes, methods and inheritance for the program.

You do not need to write the get and set methods.

Member
FirstName : STRING
LastName : STRING
DateOfBirth : DATE
Gender : STRING
Constructor()
Introduction()
DisplayFullscreenAndDateOfBirth()

Team
TeamName : STRING
TeamList : ARRAY OF Member
Constructor()
.....
.....

Competitor
Sport : STRING
Constructor()
Introduction()

Official
.....
.....
Constructor()
DisplayJobTitle()

[3]

- (b)** Write **program code** for the Member class.

Programming language

Program code

- (c) Write **program code** for the Competitor class.

Programming language

Program code

- (d) Omar Ellaboudy is an official at X-Games. He is first-aid trained and his job title is Judge. He is male and was born on 17/03/1993.

Write **program code** to create an instance of an object with the identifier BMXJudge. All attributes of the instance must be fully initialised.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

[3]

Question 5 begins on the next page.

- 5 A company is developing an application program. The project manager has been asked to create a work breakdown schedule for the project as follows:

Activity		Days to complete	Predecessor activity
A	Gather User Requirements	6	
B	Design work	4	A
C	Develop server code	4	B
D	Develop application code	5	B
E	User Interface Development	6	B
F	Test server code	2	C
G	Test application	2	D, E
H	Test application/server integration	6	F, G
I	Roll out mobile application	6	H

- (a) A GANTT chart is created from the work breakdown schedule. Activities **A** and **B** have already been added to the chart.

Complete the GANTT chart.

Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Day number																														
A																														
B																														
C																														
D																														
E																														
F																														
G																														
H																														
I																														

[5]

- (b) State which activities can run in parallel on the following days.

(i) Day 14

..... [1]

(ii) Day 16

..... [1]

- (c) Explain how the project manager will use the GANTT chart to make sure the project is completed on time.

.....
.....
.....
.....

[2]

- 6 An Abstract Data Type (ADT) is used to create an unordered binary tree. The binary tree is created as an array of nodes. Each node consists of a data value and two pointers.

A record type, Node, is declared using pseudocode.

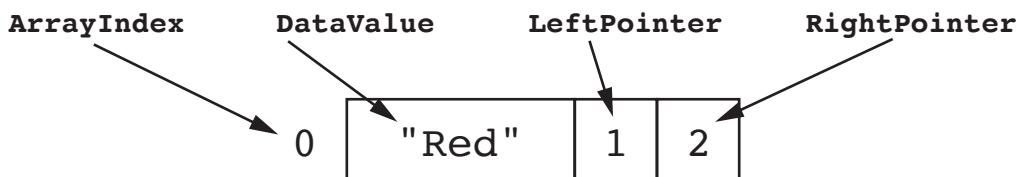
```
TYPE Node
  DECLARE DataValue : STRING
  DECLARE LeftPointer : INTEGER
  DECLARE RightPointer : INTEGER
ENDTYPE
```

The following statement declares an array BinaryTree.

```
DECLARE BinaryTree : ARRAY[0:14] OF Node
```

A variable, NextNode, points to the next free node.

The following diagram shows a possible node.



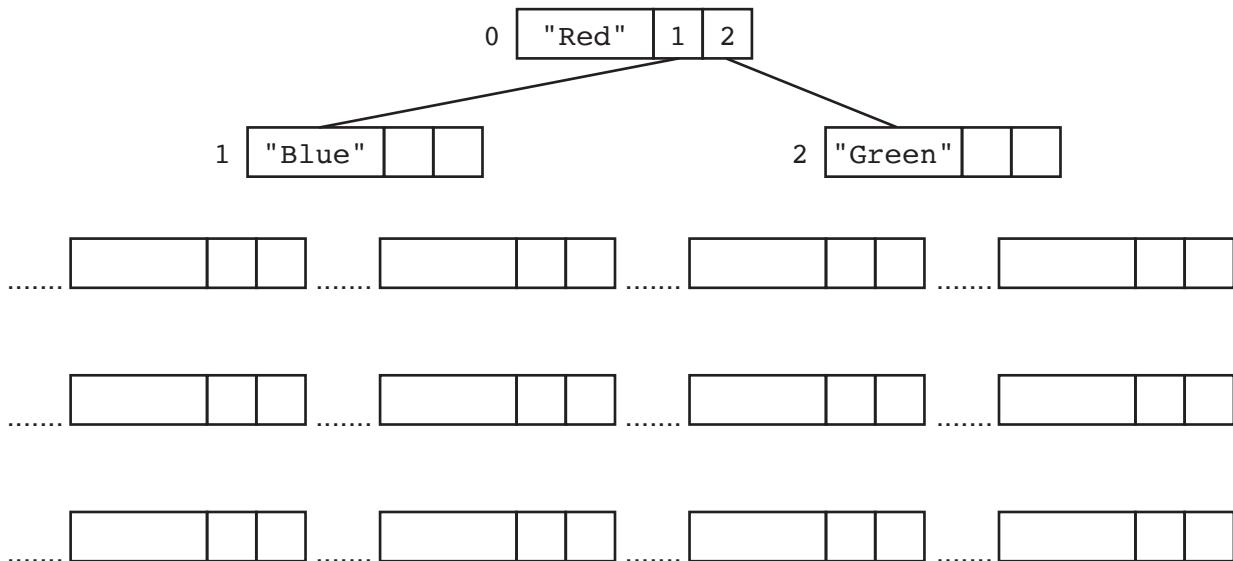
The commands in the following table create and add nodes to the binary tree.

Command	Comment
CreateTree (NodeData)	Sets NextNode to 0. Writes NodeData into DataValue at the position NextNode Updates NextNode using NextNode = NextNode + 1
AttachLeft (NodeData, ParentNode)	Writes NodeData into DataValue of NextNode Sets the LeftPointer of node ParentNode to NextNode Updates NextNode using NextNode = NextNode + 1
AttachRight (NodeData, ParentNode)	Writes NodeData into DataValue of NextNode Sets the RightPointer of node ParentNode to NextNode Updates NextNode using NextNode = NextNode + 1

- (a) The following commands are executed.

```
CreateTree("Red")
AttachLeft("Blue", 0)
AttachRight("Green", 0)
```

The following diagram shows the current state of the binary tree.



Write on the diagram to show the state of the binary tree after the following commands have been executed.

```
AttachRight("Black", 2)
AttachLeft("Brown", 2)
AttachLeft("Peach", 3)
AttachLeft("Yellow", 1)
AttachRight("Purple", 1)
AttachLeft("White", 6)
AttachLeft("Pink", 7)
AttachLeft("Grey", 9)
AttachRight("Orange", 9)
```

[5]

- (b) A new command has been added to initialise the pointers of the binary tree to -1 to indicate they are not in use.

A leaf is a node of the binary tree which has no children. In the case of this binary tree, a node with a `LeftPointer` of -1 and a `RightPointer` of -1 is a leaf.

Write a **recursive** function, in **program code**, to traverse the binary tree and output the value of **DataValue** for each leaf node.

Programming language

Program code

[8]

.[8]

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/41

Paper 4 Practical

October/November 2021

2 hours 30 minutes

You will need: Candidate source files (listed on page 2)
evidence.doc



INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the document **evidence.doc**.

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

A source file is used to answer question **2(e)**. The file is called `Pictures.txt`

1 Study the following pseudocode for a recursive function.

```
FUNCTION Unknown (BYVAL X, BYVAL Y : INTEGER) RETURNS INTEGER

IF X < Y THEN

    OUTPUT X + Y

    RETURN (Unknown(X + 1, Y) * 2)

ELSE

    IF X = Y THEN

        RETURN 1

    ELSE

        OUTPUT X + Y

        RETURN (Unknown(X - 1, Y) DIV 2)

    ENDIF

ENDIF

ENDFUNCTION
```

The operator `DIV` returns the integer value after division e.g. `13 DIV 2` would give `6`

(a) Write program code to declare the function `Unknown()`.

Save your program as **question 1**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[3]

- (b) The main program needs to run all **three** of the following function calls and output the result of each call:

Unknown (10, 15)
 Unknown (10, 10)
 Unknown (15, 10)

- (i) For each of the **three** function calls, the main program needs to:

- output the value of the two parameters
- call the function with those parameters
- output the return value.

Write the program code for the main program.

Save your program.

Copy and paste the program code into **part 1(b)(i)** in the evidence document.

[3]

- (ii) Take a screenshot to show the output from **part (b)(i)**.

Copy and paste the screenshot into **part 1(b)(ii)** in the evidence document.

[2]

- (c) Rewrite the function `Unknown ()` as an iterative function, `IterativeUnknown ()`.

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[7]

- (d) The iterative function needs to be called **three** times with the same parameters as in **part (b)**.

- (i) For each of the **three** function calls, the main program needs to:

- output the value of the two parameters
- call the iterative function with those parameters
- output the return value.

Amend the main program to perform these tasks.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[1]

- (ii) Take one or more screenshots to show the output of both functions for each set of parameters.

Copy and paste the screenshot(s) into **part 1(d)(ii)** in the evidence document.

[1]

- 2 A program, written using object-oriented programming, stores pictures as objects.

The program stores the dimensions of the picture (width and height), the colour of the frame (e.g. black), and a description of the picture (e.g. flowers).

The class has the following attributes and methods.

Picture	
Description : STRING Width : INTEGER Height : INTEGER FrameColour : STRING	// stores a description of the picture // stores the width e.g. 30 // stores the height e.g. 40 // stores the colour e.g. black
Constructor() GetDescription() GetHeight() GetWidth() GetColour() SetDescription()	// takes all four values as parameters and sets them to the private attributes // returns the description of the picture // returns the height // returns the width // returns the frame colour // takes the new description as a parameter and writes the value to description

- (a) The constructor takes the picture description, frame colour, height, and width as parameters and sets these to the private attributes.

Write the program code to declare the class `Picture` and its constructor.
Do not write any other methods.

Use your language appropriate constructor. All attributes should be private.

If you are writing in Python programming language, include attribute declarations using comments.

Save your program as **question 2**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[5]

- (b) The four get methods return the associated attribute, for example, `GetDescription()` returns the description of the picture.

Write the **four** get methods.

Save your program.

Copy and paste the program code into **part 2(b)** in the evidence document.

[3]

- (c) The method `SetDescription()` takes a new description as a parameter, and writes this value to the appropriate attribute.

Write the method `SetDescription()`.

Save your program.

Copy and paste the program code into **part 2(c)** in the evidence document.

[2]

- (d) Write program code to declare an array of type `Picture` with 100 elements.

Save your program.

Copy and paste the program code into **part 2(d)** in the evidence document.

[1]

- (e) The text file `Pictures.txt` stores the data for the pictures in the order: description, width, height, colour.

For example, for the first picture in the text file:

Flowers is the description
45 is the width
50 is the height
black is the frame colour.

The data read into the program from the text file is stored in an array of type `Picture`.
The main program and the function will need to access the array data.

The function `ReadData()`:

- opens the file `Pictures.txt`
- reads the data from the file
- creates a new object of type `Picture` for each picture
- writes each object to the array
- raises an exception if the file cannot be found
- counts and returns the number of pictures in the array.

Write program code for the function `ReadData()`.

Save your program.

Copy and paste the program code into **part 2(e)** in the evidence document.

[8]

- (f) The main program calls the function `ReadData()`.

Write the main program.

Save your program.

Copy and paste the program code into **part 2(f)** in the evidence document.

[2]

- (g) The main program needs to ask the user to input their requirements for a picture. The user will enter the colour of the frame, the maximum width, and the maximum height of the picture.

The program will then search the array of pictures, and output the picture description, the width, and the height of any picture that meets the user's requirements.

The program should allow the user to input the colour in any case (e.g. Silver, silver, or SILVER), and still output the correct results.

Edit the main program to perform the described actions.

Save your program.

Copy and paste the program code into **part 2(g)** in the evidence document.

[7]

- (h) Test your program by inputting the following search criteria:

- BLACK, 100, 100
- silver, 25, 25

Take screenshots to show the output for both search criteria.

Copy and paste the screenshots into **part 2(h)** in the evidence document.

[2]

BLANK PAGE

- 3 An ordered binary tree stores integer data in ascending numerical order.

The data for the binary tree is stored in a 2D array with the following structure:

Index	LeftPointer	Data	RightPointer
[0]	[0]	[1]	[2]
[0]	1	10	2
[1]	-1	5	-1
[2]	-1	16	-1

Each row in the table represents one node on the tree.

The number -1 represents a null pointer.

- (a) The 2D array, `ArrayNodes`, is declared with space for 20 nodes.

Each node has a left pointer, data and right pointer.

The program also initialises the:

- `RootPointer` to -1 (null); this points to the first node in the binary tree
- `FreeNode` to 0; this points to the first empty node in the array.

Write program code to declare `ArrayNodes`, `RootPointer` and `FreeNode` in the main program.

If you are writing in Python programming language, include attribute declarations using comments.

Save your program as **question 3**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[4]

- (b) The procedure `AddNode()` adds a new node to the array `ArrayNodes`.

The procedure needs to:

- take the array, root pointer and free node pointer as parameters
- ask the user to enter the data and read this in
- add the node to the root pointer if the tree is empty
- otherwise, follow the pointers to find the position for the data item to be added
- store the data in the location and update all pointers.

There are **six** incomplete statements in the following pseudocode for the procedure AddNode () .

```

PROCEDURE AddNode (BYREF ArrayNodes[] : ARRAY OF INTEGER,
                  BYREF RootPointer : INTEGER, BYREF FreeNode : INTEGER)
  OUTPUT "Enter the data"
  INPUT NodeData
  IF FreeNode <= 19 THEN
    ArrayNodes[FreeNode, 0] ← -1
    ArrayNodes[FreeNode, 1] ← .....
    ArrayNodes[FreeNode, 2] ← -1
    IF RootPointer = ..... THEN
      RootPointer ← 0
    ELSE
      Placed ← FALSE
      CurrentNode ← RootPointer
      WHILE Placed = FALSE
        IF NodeData < ArrayNodes[CurrentNode, 1] THEN
          IF ArrayNodes[CurrentNode, 0] = -1 THEN
            ArrayNodes[CurrentNode, 0] ← .....
            Placed ← TRUE
          ELSE
            ..... ← ArrayNodes[CurrentNode, 0]
          ENDIF
        ELSE
          IF ArrayNodes[CurrentNode, 2] = -1 THEN
            ArrayNodes[CurrentNode, 2] ← FreeNode
            Placed ← .....
          ELSE
            CurrentNode ← ArrayNodes[CurrentNode, 2]
          ENDIF
        ENDIF
      ENDIF
      ENDWHILE
    ENDIF
    FreeNode ← ..... + 1
  ELSE
    OUTPUT("Tree is full")
  ENDIF
ENDPROCEDURE

```

Write **program code** for the procedure AddNode () .

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[8]

- (c) The procedure `PrintAll()` outputs the data in each element in `ArrayNodes`, in the order they are stored in the array.

Each element is printed in a row in the order:

LeftPointer Data RightPointer

For example:

1	20	-1
-1	10	-1

Write program code for the procedure `PrintAll()`.

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[4]

- (d) The main program should loop 10 times, each time calling the procedure `AddNode()`. It should then call the procedure `PrintAll()`.

(i) Edit the main program to perform the actions described.

Save your program.

Copy and paste the program code into **part 3(d)(i)** in the evidence document.

[3]

(ii) Test the program by entering the data:

10
5
15
8
12
6
20
11
9
4

Take a screenshot to show the output after the given data are entered.

Copy and paste the screenshot into **part 3(d)(ii)** in the evidence document.

[1]

- (e) An in-order tree traversal visits the left node, then the root (and outputs this), then visits the right node.
- (i) Write a recursive procedure, `InOrder()`, to perform an in-order traversal on the tree held in `ArrayNodes`.

Save your program.

Copy and paste the program code into **part 3(e)(i)** in the evidence document.

[7]

- (ii) Test the procedure `InOrder()` with the same data entered in **part (d)(ii)**.

Take a screenshot to show the output after entering the data.

Copy and paste the screenshot into **part 3(e)(ii)** in the evidence document.

[1]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/43

Paper 4 Further Problem-solving and Programming Skills

October/November 2018

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

- 1 A declarative language is used to represent the following facts and rules about animals.

```

01 feature(dog, drinks_milk).

02 feature(dog, has_lungs).

03 feature(horse, has_lungs).

04 feature(tuna, lives_in_water).

05 feature(tuna, has_gills).

06 feature(crab, lives_in_water).

07 mammal(drinks_milk).

08 mammal(has_lungs).

09 fish(lives_in_water).

10 fish(has_gills).

11 is_a_mammal(X) IF (feature(X, Y) AND mammal(Y)) AND (feature(X, Z)
AND mammal(Z)).

```

These clauses are explained in the following table.

Clause	Explanation
01	A dog has the feature, drinks milk
07	A mammal drinks milk
11	X is a mammal, if: • X has the feature Y and a mammal has a feature Y, and • X has the feature Z and a mammal has the feature Z

- (a) More facts are to be included.

- (i) A bird has wings, and a bird lays eggs.

Write the additional clauses to record these facts.

12

13

[2]

- (ii) An eagle has all the features of a bird.

Write the additional clauses to record this fact.

14

15

[2]

- (b) (i) Using the variable B, the goal

```
feature(B, drinks_milk)
```

returns

B = dog

Write the result returned by the goal

```
feature(B, lives_in_water)
```

B = [2]

- (ii) Write a goal, using the variable C, to find the feature(s) of tuna.

..... [2]

- (c) An animal is a bird if it lays eggs **and** it has wings.

Complete the following rule.

is_a_bird(X) IF

..... [3]

- (d) Declarative programming and object-oriented programming are two examples of programming paradigms.

- (i) Define the term **programming paradigm**.

.....

..... [1]

- (ii) Give **two** examples of programming paradigms, other than declarative and object-oriented programming.

1

2

[2]

- 2 Kendra collects books. She is writing a program to store and analyse information about her books.

Her program stores information about each book as a record. The following table shows the information that will be stored about each book.

Field name	Description
Title	The title of the book
Author	The first listed author of the book
ISBN	A 13-digit code that uniquely identifies the book, for example: "0081107546738"
Fiction	If the book is fiction (TRUE) or non-fiction (FALSE)
LastRead	The date when Kendra last read the book

- (a) Write **pseudocode** to declare an Abstract Data Type (ADT) named Book, to store the information in the table.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[4]

- (b)** The records are stored in a random access file.

The function, `Hash()`, takes as a parameter the ISBN and returns the hash value.

The disk address of the record in the hash table is calculated as: ISBN modulus 2000 plus 1.

Write program code for the function Hash () .

Programming language

Program code

[4]

[4]

- (c) The random access file, MyBooks.dat, stores the data about the books in the format:

```
<Title>
<Author>
<ISBN>
<Fiction>
<LastRead>
```

A procedure, FindBook():

- prompts the user to input the ISBN of a book until the ISBN contains 13 numeric digits
 - uses the function `Hash()` to calculate the disk address of the record
 - reads the record for that book from `MyBooks.dat` into a variable of type `Book`
 - outputs all the data about the book.

Use **pseudocode** to write the procedure `FindBook()`.

You can assume that the record exists at the disk address generated.

. [8]

- 3 Joseph is taking a toy apart. Each time he removes an item from the toy, he writes the name of the item at the bottom of a paper list. When he rebuilds the toy, he puts the items back together working from the bottom of the list.

Joseph writes a computer program to create the list using a stack, `Parts`.

- (a) Describe a stack structure.

.....
..... [1]

- (b) The stack is represented as an array in the program, the first element in the array is [0].

The current contents of the stack, `Parts`, and its pointer, `StackPointer` are shown.

StackPointer	<input type="text" value="5"/>	StackContents
0		"Screw 1"
1		"Screw 2"
2		"Back case"
3		"Screw 3"
4		"Engine outer"
5		
6		
7		

- (i) Describe the purpose of the variable `StackPointer`.

.....
..... [1]

- (ii) The procedure `POP()` removes an item from the stack. The procedure `PUSH(<identifier>)` adds an item to the stack.

The current contents of the stack, `Parts`, and its pointer, `StackPointer` are shown.

StackPointer	<input type="text" value="5"/>	StackContents
0		"Screw 1"
1		"Screw 2"
2		"Back case"
3		"Screw 3"
4		"Engine outer"
5		
6		
7		

Use the table below to show the contents of the stack, `Parts`, and its pointer after the following code is run.

```

POP()
POP()
PUSH("Light 1")
PUSH("Light 2")
PUSH("Wheel 1")
POP()
POP()

```

StackPointer	<input type="text"/>	StackContents
0		
1		
2		
3		
4		
5		
6		
7		

- (c) A 1D array, Parts, is used to implement the stack. Parts is declared as:

```
DECLARE Parts : ARRAY[0 : 19] OF STRING
```

- (i) The procedure POP outputs the last element that has been pushed onto the stack and replaces it with a '*'.

Complete the **pseudocode** for the procedure POP.

```
PROCEDURE POP
```

```
IF ..... = .....
```

```
THEN
```

```
OUTPUT "The stack is empty"
```

```
ELSE
```

```
StackPointer ← .....
```

```
OUTPUT .....
```

```
Parts[StackPointer] ← .....
```

```
ENDIF
```

```
ENDPROCEDURE
```

[5]

- (ii) The procedure PUSH() puts the parameter onto the stack.

Complete the **pseudocode** for the procedure PUSH().

```
PROCEDURE PUSH(BYVALUE Value : String)
```

```
IF StackPointer > .....
```

```
THEN
```

```
OUTPUT "Stack full"
```

```
ELSE
```

```
..... ← .....
```

```
StackPointer ← .....
```

```
ENDIF
```

```
ENDPROCEDURE
```

[4]

- 4 The recursive algorithm for the `Calculate()` function is defined as follows:

```
01 FUNCTION Calculate(BYVALUE Number : INTEGER) RETURNS INTEGER  
02     IF Number = 0  
03         THEN  
04             Calculate ← -10  
05         ELSE  
06             Calculate ← Number * Calculate(Number - 1)  
07     ENDIF  
08 ENDFUNCTION
```

- (a) (i) State what is meant by a **recursive algorithm**.

..... [1]

- (ii) State the line number in `Calculate()` where the recursive call takes place.

..... [1]

Question 4(b) begins on the next page.

- (b) The function is called with Calculate(3).

Dry run the function **and** complete the trace table below. State the final value returned. Show your working.

```

01  FUNCTION Calculate(BYVALUE Number : INTEGER) RETURNS INTEGER
02      IF Number = 0
03          THEN
04              Calculate ← -10
05          ELSE
06              Calculate ← Number * Calculate(Number - 1)
07      ENDIF
08  ENDFUNCTION

```

Working

.....

.....

.....

Trace table:

Call number	Function call	Number = 0 ?	Return value

Final return value

[6]

- (c) A recursive algorithm within a subroutine can be replaced with an iterative algorithm.
- (i) Describe **one** problem that can occur when running a subroutine that has a recursive algorithm.

.....
.....
.....
..... [2]

- (ii) Rewrite the Calculate() function in **pseudocode**, using an **iterative algorithm**.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [5]

- 5 A game uses a set of cards. Each card has a number (between 0 and 9 inclusive) and a shape ("square", "triangle" or "circle").

The game is written using object-oriented programming.

The class, Cards, has the private properties:

- Number
- Shape

and the methods:

- Constructor()
- GetNumber()
- GetShape()

The purpose of each method in the class Cards is given in the following table.

Method	Purpose
Constructor()	Takes a number and a shape as parameters Checks that the number and the shape are valid and: <ul style="list-style-type: none"> • either assigns the parameters to Number and Shape • or reports an error.
GetNumber()	A public method that returns the number for that card.
GetShape()	A public method that returns the shape for that card.

- (a) Explain why the properties are private.

.....

[2]

- (b)** Write **program code** for the `Constructor()` method.

Programming language

Program code

[5]

[5]

- (c) Write program code for the GetNumber() method.**

Programming language

Program code

[2]

[2]

- (d) A card, Ones, has the value 1 for Number and the value "square" for Shape.

Write program code to instantiate an instance of Cards for Ones

Programming language

Program code

[2]

[2]

- (e) The game has a function, `Compare()` that takes two cards as parameters and compares them.

If the cards are identical, the function outputs "SNAP" and returns -1. If they are not identical, and the card numbers are different, it returns the Number of the card with the higher value or the Number for the cards if they are the same.

Write program code for the Compare () function.

Programming language

Program code

[6]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/41

Paper 4 Further Problem-solving and Programming Skills

May/June 2015

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.

Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language

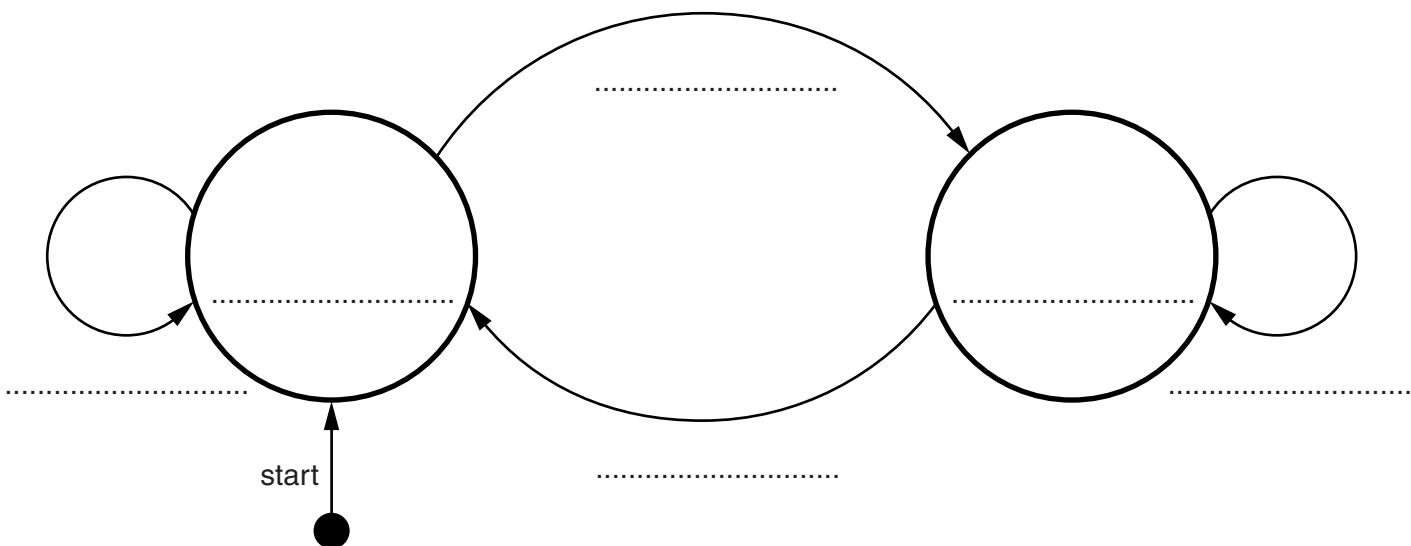
- 1 A turnstile is a gate which is in a locked state. To open it and pass through, a customer inserts a coin into a slot on the turnstile. The turnstile then unlocks and allows the customer to push the turnstile and pass through the gate.

After the customer has passed through, the turnstile locks again. If a customer pushes the turnstile while it is in the locked state, it will remain locked until another coin is inserted.

The turnstile has two possible states: **locked** and **unlocked**. The transition from one state to another is as shown in the table below.

Current state	Event	Next state
Locked	Insert coin	Unlocked
Locked	Push	Locked
Unlocked	Attempt to insert coin	Unlocked
Unlocked	Pass through	Locked

Complete the state transition diagram for the turnstile:



[5]

- 2** A declarative programming language is used to represent the knowledge base shown below:

```

01 capital_city(amman).
02 capital_city(beijing).
03 capital_city(brussels).
04 capital_city(cairo).
05 capital_city(london).
06 city_in_country(amman, jordan).
07 city_in_country(shanghai, china).
08 city_in_country(brussels, belgium).
09 city_in_country(london, uk).
10 city_in_country(manchester, uk).
11 country_in_continent(belgium, europe).
12 country_in_continent(china, asia).
13 country_in_continent(uk, europe).
14 city_visited(amman).
15 city_visited(beijing).
16 city_visited(cairo).

```

These clauses have the following meaning:

Clause	Explanation
01	Amman is a capital city
06	Amman is a city in the country of Jordan
11	Belgium is a country in the continent of Europe
14	The travel writer visited Amman

- (a)** More facts are to be included.

The travel writer visited the city of Santiago which is the capital city of Chile, in the continent of South America.

Write additional clauses to record this.

17

.....

18

.....

19

.....

20

.....

[4]

- (b) Using the variable ThisCountry, the goal

```
country_in_continent(ThisCountry, europe)
```

returns

```
ThisCountry = belgium, uk
```

Write the result returned by the goal:

```
city_in_country(ThisCity, uk)
```

ThisCity = [2]

- (c) Complete the rule below to list the countries the travel writer has visited.

```
countries_visited(ThisCountry)
```

IF [4]

.....

.....

.....

..... [4]

- 3 A shop gives some customers a discount on goods totalling more than \$20.
 The discounts are:
- 5% for goods totalling more than \$100
 - 5% with a discount card
 - 10% with a discount card and goods totalling more than \$100

(a) Complete the decision table.

Conditions	goods totalling more than \$20	Y	Y	Y	Y	N	N	N	N
	goods totalling more than \$100	Y	Y	N	N	Y	Y	N	N
	have discount card	Y	N	Y	N	Y	N	Y	N
Actions	No discount								
	5% discount								
	10% discount								

[4]

(b) Simplify your solution by removing redundancies.

Conditions	goods totalling more than \$20								
	goods totalling more than \$100								
	have discount card								
Actions	No discount								
	5% discount								
	10% discount								

[5]

- (c) The simplified table produced in part (b) is used as a design for program code.

The following identifier table shows the parameters to be passed to the function `Discount`. This function returns the discount amount as an integer.

Identifier	Data type
GoodsTotal	INTEGER
HasDiscountCard	BOOLEAN

Write program code for this function.

Programming language

[6]

- 4 A payroll program is to be written using an object-oriented programming language. An `Employee` class is designed. Two subclasses have been identified:

 - `HourlyPaidEmployee` who is paid a monthly wage calculated from their hourly rate of pay and the number of hours worked during the month
 - `SalariedEmployee` who is paid a monthly wage which is one 12th of their annual salary

(a) Draw an inheritance diagram for these classes.

[3]

- (b)** The design for the Employee class consists of:

 - properties
 - EmployeeName
 - EmployeeID
 - AmountPaidThisMonth
 - methods
 - SetEmployeeName
 - SetEmployeeID
 - CalculatePay

Write **program code** for the class definition of the superclass Employee.

Programming language

.....

.....

.....

.....

.....

.....

.....

.....

.....

[5]

- (c) (i) State the properties and/or methods required for the subclass HourlyPaidEmployee.

.....
.....
.....
..... [4]

- (ii) State the properties and/or methods required for the subclass SalariedEmployee.

.....
.....
.....
..... [2]

- (d) Name the feature of object-oriented program design that allows the method CalculatePay to be declared in the superclass Employee.

.....
..... [1]

- 5 Data is stored in the array `NameList[1:10]`. This data is to be sorted.

- (a) (i) Complete the pseudocode algorithm for an insertion sort.

```

FOR ThisPointer ← 2 TO .....
    // use a temporary variable to store item which is to
    // be inserted into its correct location
    Temp ← NameList[ThisPointer]
    Pointer ← ThisPointer - 1

    WHILE (NameList[Pointer] > Temp) AND .....
        // move list item to next location
        NameList[.....] ← NameList[.....]
        Pointer ← .....
    ENDWHILE

    // insert value of Temp in correct location
    NameList[.....] ← .....

ENDFOR

```

[7]

- (ii) A special case is when `NameList` is already in order. The algorithm in part (a)(i) is applied to this special case.

Explain how many iterations are carried out for each of the loops.

.....

 [3]

- (b) An alternative sort algorithm is a bubble sort:

```
FOR ThisPointer ← 1 TO 9
    FOR Pointer ← 1 TO 9
        IF NameList[Pointer] > NameList[Pointer + 1]
            THEN
                Temp ← NameList[Pointer]
                NameList[Pointer] ← NameList[Pointer + 1]
                NameList[Pointer + 1] ← Temp
            ENDIF
        ENDFOR
    ENDFOR
```

- (i) As in part (a)(ii), a special case is when NameList is already in order. The algorithm in part (b) is applied to this special case.

Explain how many iterations are carried out for each of the loops.

.....
.....
.....
.....

[2]

- (ii) Rewrite the algorithm in part (b), using **pseudocode**, to reduce the number of unnecessary comparisons. Use the same variable names where appropriate.

[5]

- 6 A queue Abstract Data Type (ADT) has these associated operations:

- create queue
- add item to queue
- remove item from queue

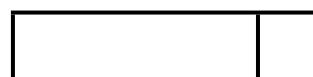
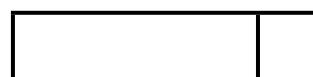
The queue ADT is to be implemented as a linked list of nodes.

Each node consists of data and a pointer to the next node.

- (a) The following operations are carried out:

```
CreateQueue  
AddName ("Ali")  
AddName ("Jack")  
AddName ("Ben")  
AddName ("Ahmed")  
RemoveName  
AddName ("Jatinder")  
RemoveName
```

Add appropriate labels to the diagram to show the final state of the queue. Use the space on the left as a workspace. Show your final answer in the node shapes on the right:



[3]

- (b) Using pseudocode, a record type, Node, is declared as follows:

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

The statement

```
DECLARE Queue : ARRAY[1:10] OF Node
```

reserves space for 10 nodes in array Queue.

- (i) The CreateQueue operation links all nodes and initialises the three pointers that need to be used: HeadPointer, TailPointer and FreePointer.

Complete the diagram to show the value of all pointers after CreateQueue has been executed.

Queue		
	Name	Pointer
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		
[10]		

HeadPointer

TailPointer

FreePointer

[4]

- (ii) The algorithm for adding a name to the queue is written, using pseudocode, as a procedure with the header:

```
PROCEDURE AddName (NewName)
```

where NewName is the new name to be added to the queue.

The procedure uses the variables as shown in the identifier table.

Identifier	Data type	Description
Queue	Array[1:10] OF Node	Array to store node data
NewName	STRING	Name to be added
FreePointer	INTEGER	Pointer to next free node in array
HeadPointer	INTEGER	Pointer to first node in queue
TailPointer	INTEGER	Pointer to last node in queue
CurrentPointer	INTEGER	Pointer to current node

```
PROCEDURE AddName (BYVALUE NewName : STRING)
    // Report error if no free nodes remaining
    IF FreePointer = 0
        THEN
            Report Error
    ELSE
        // new name placed in node at head of free list
        CurrentPointer ← FreePointer
        Queue[CurrentPointer].Name ← NewName
        // adjust free pointer
        FreePointer ← Queue[CurrentPointer].Pointer
        // if first name in queue then adjust head pointer
        IF HeadPointer = 0
            THEN
                HeadPointer ← CurrentPointer
        ENDIF
        // current node is new end of queue
        Queue[CurrentPointer].Pointer ← 0
        TailPointer ← CurrentPointer
    ENDIF
ENDPROCEDURE
```

Complete the **pseudocode** for the procedure RemoveName. Use the variables listed in the identifier table.

```
PROCEDURE RemoveName ()  
    // Report error if Queue is empty  
  
    .....  
  
    .....  
  
    .....  
  
    OUTPUT Queue[.....] .Name  
  
    // current node is head of queue  
  
    .....  
  
    // update head pointer  
  
    .....  
  
    // if only one element in queue then update tail pointer  
  
    .....  
  
    .....  
  
    .....  
  
    .....  
  
    // link released node to free list  
  
    .....  
  
    .....  
  
    .....  
  
ENDPROCEDURE
```

[6]

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/42

Paper 4 Practical

May/June 2022

2 hours 30 minutes



You will need: Candidate source files (listed on page 2)
evidence.doc

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is not saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
 - Java (console mode)
 - Python (console mode)
 - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [].

This document has **12** pages. Any blank pages are indicated.

Open the document **evidence.doc**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

If the programming language used does not support arrays, a list can be used instead.

A source file is used to answer **Question 3**. The file is called **CardValues.txt**

- 1 A program needs to use a stack data structure. The stack can store up to 10 integer elements.

A 1D array `StackData` is used to store the stack globally. The global variable `StackPointer` points to the next available space in the stack and is initialised to 0.

- (a) Write program code to declare the array and pointer as global data structures. Initialise the pointer to 0.

Save your program as **Question1_J22**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[3]

- (b) Write a procedure to output all 10 elements in the stack **and** the value of `StackPointer`.

Save your program.

Copy and paste the program code into **part 1(b)** in the evidence document.

[3]

- (c) The function `Push()` takes an integer parameter and returns `FALSE` if the stack is full. If the stack is not full, it puts the parameter value on the stack, updates the relevant pointer and returns `TRUE`.

Write program code for the function `Push()`.

Save your program.

Copy and paste the program code into **part 1(c)** in the evidence document.

[6]

(d) (i) Edit the main program to test the `Push()` function. The main program needs to:

- allow the user to enter 11 numbers and attempt to add these to the stack
- output an appropriate message when a number is added to the stack
- output an appropriate message when a number is not added to the stack if it is full
- output the contents of the stack after attempting to add all 11 numbers.

Save your program.

Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[5]

(ii) Test your program from **part 1(d)(i)** with the following 11 inputs:

11 12 13 14 15 16 17 18 19 20 21

Take a screenshot to show the output.

Copy and paste the screenshot into **part 1(d)(ii)** in the evidence document.

[1]

(e) The function `Pop()` returns `-1` if the stack is empty. If the stack is not empty, it returns the element at the top of the stack and updates the relevant pointer.

(i) Write program code for the function `Pop()`.

Save your program.

Copy and paste the program code into **part 1(e)(i)** in the evidence document.

[5]

(ii) After the code you wrote in the main program for **part 1(d)(i)**, add program code to:

- remove two elements from the stack using `Pop()`
- output the updated contents of the stack.

Test your program and take a screenshot to show the output.

Copy and paste the screenshot into **part 1(e)(ii)** in the evidence document.

[2]

2 A 2D array stores data entered by a user.

- (a) The main program declares a 2D array of 10 by 10 integer elements.

The array is initialised with a random number between 1 and 100 in each element.

Write program code for the main program.

Save your program as **Question2_J22**.

Copy and paste the program code into **part 2(a)** in the evidence document.

[4]

- (b) The following bubble sort pseudocode algorithm sorts the data in the first dimension of the 2D array into ascending numerical order.

```

ArrayLength ← 10

FOR X ← 0 TO ArrayLength - 1

    FOR Y ← 0 TO ArrayLength - 2

        FOR Z ← 0 TO ArrayLength - Y - 2

            IF ArrayData[X, Z] > ArrayData[X, Z + 1] THEN

                TempValue ← ArrayData[X, Z]

                ArrayData[X, Z] ← ArrayData[X, Z+1]

                ArrayData[X, Z + 1] ← TempValue

            ENDIF

        NEXT Z

    NEXT Y

NEXT X

```

- (i) Amend your main program by writing program code to implement the bubble sort algorithm after the initialisation of the array elements.

You must **not** use any built-in sorting functions for your programming language.

Save your program.

Copy and paste the program code into **part 2(b)(i)** in the evidence document.

[5]

- (ii) Write program code for a procedure to output all the values in the 2D array. The values should be output as a 2D grid, with values in rows and columns.

Call the procedure before and after your bubble sort code.

Save your program.

Copy and paste the program code into **part 2(b)(ii)** in the evidence document.

[3]

- (iii) Test your program.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 2(b)(iii)** in the evidence document.

[1]

- (c) The following pseudocode function uses recursion to perform a binary search in the first row of the array, for the value `SearchValue` in the array `SearchArray`.

The function returns `-1` if the item was not found, or it returns the index where it is found.

There are **six** incomplete statements.

```
FUNCTION BinarySearch(SearchArray, Lower, Upper, SearchValue) RETURNS
    INTEGER

    IF Upper >= Lower THEN
        Mid ← (Lower + (Upper - 1)) DIV .....
        IF SearchArray[0, Mid] = ..... THEN
            RETURN .....
        ELSE
            IF SearchArray[0, Mid] > SearchValue THEN
                RETURN BinarySearch(SearchArray, ..... , Mid - 1,
                                     SearchValue)
            ELSE
                RETURN BinarySearch(SearchArray, Mid + 1, ..... ,
                                     SearchValue)
            ENDIF
        ENDIF
    ENDIF
    RETURN .....
ENDFUNCTION
```

Note: the arithmetic operator `DIV` performs integer division, e.g. the result of `10 DIV 3` will be `3`.

- (i) Write program code for the recursive function `BinarySearch()`.

Save your program.

Copy and paste the program code into **part 2(c)(i)** in the evidence document.

[8]

- (ii) In the main program, test the function `BinarySearch()` twice, outputting the returned value each time.

One test should be for a number that is in the first line of the array.
One test should be for a number that is not in the first line of the array.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 2(c)(ii)** in the evidence document.

[2]

- 3 A programmer is designing a computer game that uses a set of cards.

Each card has a number and a colour. The cards are saved in the text file `CardValues.txt`

The program has a class named `Card`. The class has the following attributes and methods.

Card	
Number : INTEGER	The number of the card
Colour : STRING	The colour of the card
Constructor()	Takes two values as parameters and sets them to the private attributes
GetNumber()	Returns the number of the card
GetColour()	Returns the colour of the card

- (a) The constructor takes the number and colour of the card as parameters and sets them to the private attributes.

Write program code to declare the class `Card` and its constructor. Do **not** write any other methods.

Use your programming language appropriate constructor.

All attributes should be private. If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_J22**.

Copy and paste the program code into **part 3(a)** in the evidence document.

[5]

- (b) The two get methods return the associated attribute.

Write program code for the get methods `GetNumber()` and `GetColour()`.

Save your program.

Copy and paste the program code into **part 3(b)** in the evidence document.

[3]

- (c) The text file `CardValues.txt` stores the data for 30 cards, in the order: number, colour.

For example, the first card in the text file:

1 is the number
red is the colour.

A 1D array of type `Card` is declared to store all the cards read in from `CardValues.txt`

Write the main program to:

- declare an array of type `Card` with 30 elements
- read in the data for the 30 cards from `CardValues.txt` and assign each to the array.

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[7]

- (d) The program needs to allow all players (maximum of 5) to select 4 cards from the 30 available. A card can only be selected once, so the program needs to record which cards have already been selected.

The function, `ChooseCard()`:

- takes as input an integer to represent an array index from 1 to 30
- validates that the value is between 1 and 30 inclusive
- checks if the card is available (it has not already been selected)
- loops until an available card is selected
- returns the index of the card if it is available.

Amend the program to store which cards have already been selected **and** write program code for the function `ChooseCard()`.

Save your program.

Copy and paste the program code into **part 3(d)** in the evidence document.

[6]

- (e) The main program needs to allow one player to select all their 4 cards.

(i) Amend the main program to:

- create an array, `Player1`, for player 1 of type `Card`
- ask player 1 to input 4 integers using the function from **part 3(d)**
- store the cards in `Player1`
- output the number and colour of the 4 cards in `Player1`.

Save your program.

Copy and paste the program code into **part 3(e)(i)** in the evidence document.

[5]

- (ii) Test your program with the following test data:

Test 1: 1 5 9 10

Test 2: 2 2 3 4 4 5

Take a screenshot to show the output.

Copy and paste the screenshot into **part 3(e)(ii)** in the evidence document.

[1]

BLANK PAGE

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.

CANDIDATE
NAME

CENTRE
NUMBER

--	--	--	--	--

CANDIDATE
NUMBER

--	--	--	--



COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

May/June 2019

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.

You may use an HB pencil for any diagrams, graphs or rough working.

Do not use staples, paper clips, glue or correction fluid.

DO NOT WRITE IN ANY BARCODES.

Answer **all** questions.

No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.

The number of marks is given in brackets [] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **19** printed pages and **1** blank page.

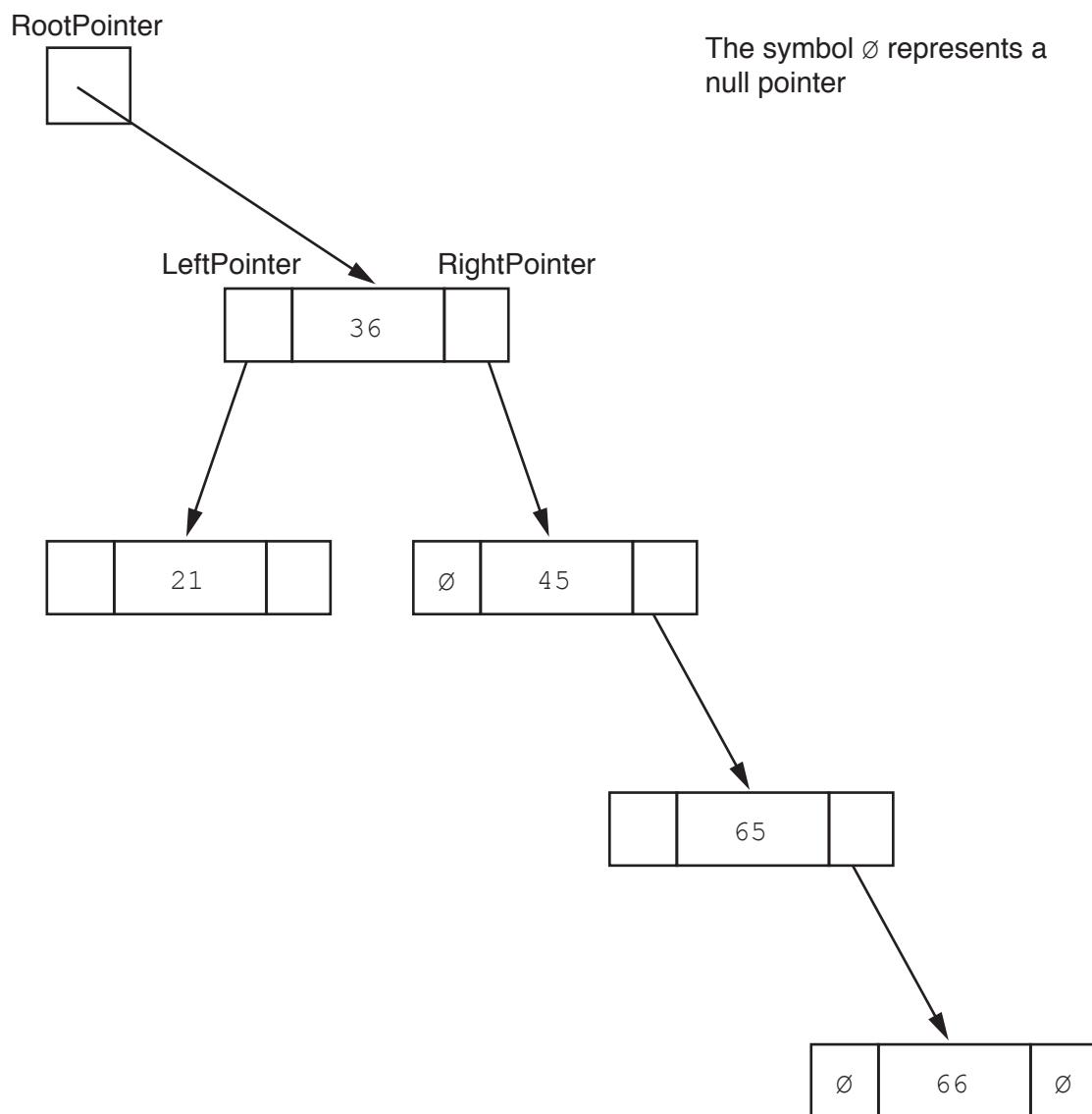
- 1 A company wants an online marking system for an examination.

- (a) The following is a selection of data showing final marks.

36, 45, 21, 65, 66, 13, 54, 53, 34

A linked list of nodes will be used to store the data. Each node consists of the data, a left pointer and a right pointer. The linked list will be organised as a binary tree.

- (i) Complete the binary tree to show how the data above will be organised.



[5]

- (ii) The following diagram shows a 2D array that stores the nodes of the binary tree's linked list.

Add the correct pointer values to complete the diagram, using your answer from part (a)(i).

RootPointer

0

Index	LeftPointer	Data	RightPointer
0		36	
1		45	
2		21	
3		65	
4		66	
5		13	
6		54	
7		53	
8		34	
9			

FreePointer

--

[6]

- (b) The company wants to implement a program for the marking system. It will do this with object-oriented programming (OOP).

Many candidates take the examination. Each examination paper is given a PaperID that is made up of the centre (school) number followed by the candidate number.

Each examination paper is awarded a grade.

The following diagram shows the design for the ExaminationPaper class. This includes the attributes and methods.

ExaminationPaper	
FinalMark : INTEGER	// maximum 2 digits, initialised to 0
Grade : STRING	// "Pass", "Merit", "Distinction" // or "Fail", initialised to "Fail"
PaperID : STRING	// centre number followed by the // candidate number, for example // "ZZ00991001"
Create()	// creates and initialises a new instance // of the ExaminationPaper class using // language-appropriate constructor
SetFinalMark()	// checks that the mark parameter has a // valid value, if so, assigns it to // FinalMark
SetGrade()	// sets Grade based on FinalMark
GetFinalMark()	// returns FinalMark
GetGrade()	// returns Grade
GetPaperID()	// returns PaperID

- (i) The constructor receives the centre number and candidate number as parameter values to create PaperID. Other properties are initialised as instructed in the class diagram.

Write **program code** for the Create () constructor method.

Programming language

Program code

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

[5]

- (ii) Get and set methods are used to support the security and integrity of data in object-oriented programming.

Explain how get and set methods are used to support security and integrity.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

[3]

- (iii) Write **program code** for the following three get methods.

Programming language

GetFinalMark ()

Program code

.....
.....
.....
.....

GetGrade ()

Program code

.....
.....
.....
.....

GetPaperID ()

Program code

.....
.....
.....
.....

[4]

- (iv) The method `SetFinalMark()` checks that its `INTEGER` parameter `Mark` is valid. It is then set as the final mark if it is valid. A valid mark is greater than or equal to 0 and less than or equal to 90.

If the mark is valid, the method sets the final mark and returns `TRUE`.

If the mark is not valid, the method does not set the final mark and returns `FALSE`.

Write program code for `SetFinalMark(Mark : INTEGER)`.

Programming language

Program code

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
..... [5]

- (v) Write **program code** for the method:

SetGrade(DistMark, MeritMark, PassMark : INTEGER)

Use the properties in the original class definition.

Grades are awarded as follows:

Grade	Criteria
Distinction	\geq DistMark
Merit	\geq MeritMark
Pass	\geq PassMark
Fail	$<$ PassMark

Programming language

Program code

[4]

- (vi) Emily is a candidate who has taken the examination paper. The grades are awarded as follows:

Grade	Criteria
Distinction	>= 80
Merit	>= 70
Pass	>= 55

The procedure `Main()` performs the following tasks.

- allows the centre number, candidate number and mark to be input, with suitable prompts
 - assigns an instance of `ExaminationPaper` to the variable `ThisPaper`
 - sets the mark for the object
 - sets the grade for the object
 - outputs the grade for the object

Write program code for the Main() procedure.

Programming language

Program code

- (c) The examination paper will be taken by many candidates in centres around the world.

The program stores the objects of the `ExaminationPaper` class in a file. The company has decided to use a hash table, rather than a linked list to store the objects.

Explain why a hash table is more suitable than a linked list to store the objects.

.....
.....
.....
.....
.....
.....
.....
.....

[4]

Question 2 begins on the next page.

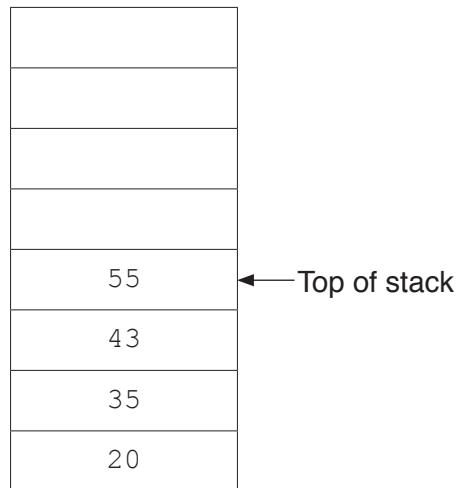
2 A stack is an Abstract Data Type (ADT).

(a) Tick (\checkmark) one box to show the statement that describes a stack data structure.

Statement	Tick (\checkmark)
Last in first out	
First in first out	
Last in last out	

[1]

(b) A stack contains the values 20, 35, 43, 55.



(i) Show the contents of the stack in part (b) after the following operations.

POP()

POP()

PUSH(10)



[1]

- (ii) Show the contents of the stack from **part (b)(i)** after these further operations:

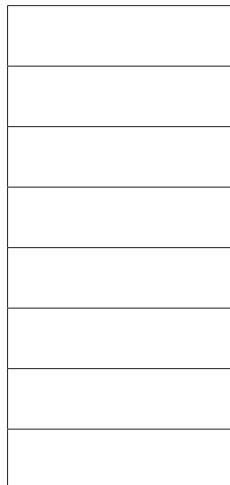
POP ()

PUSH (50)

PUSH (55)

POP ()

PUSH (65)



[1]

- (iii) The stack is implemented as a 1D array, with eight elements, and given the identifier ArrayStack.

The global variable `Top` contains the index of the last element in the stack, or `-1` if the stack is empty.

The function Push():

- takes as a parameter an `INTEGER` value to place on the stack
 - adds the value to the top of the stack and returns `TRUE` to show that the operation was successful
 - returns `FALSE` if the stack is full.

Write an algorithm in **pseudocode** for the function Push().

[7]

- 3 (a) Identify **and** describe **two** features of an editor that can help a programmer to write program code.

Feature 1

Description

.....

.....

.....

.....

.....

.....

.....

.....

.....

[4]

- (b) A programmer can use three types of test data when testing a program.

Identify the **three** different types of test data.

1

2

3

[3]

- 4 (a) A program has sorted some data in the array, List, in ascending order.

The following binary search algorithm is used to search for a value in the array.

```

01  ValueFound ← FALSE
02  UpperBound ← LengthOfList - 1
03  LowerBound ← 0
04  NotInList ← FALSE
05
06  WHILE ValueFound = FALSE AND NotInList = FALSE
07      MidPoint ← ROUND((LowerBound + UpperBound) / 2)
08
09      IF List[LowerBound] = SearchValue
10          THEN
11              ValueFound ← TRUE
12          ELSE
13              IF List[MidPoint] < SearchValue
14                  THEN
15                      UpperBound ← MidPoint + 1
16                  ELSE
17                      UpperBound ← MidPoint - 1
18                  ENDIF
19                  IF LowerBound > MidPoint
20                      THEN
21                          NotInList ← TRUE
22                      ENDIF
23                  ENDIF
24  ENDWHILE
25
26  IF ValueFound = FALSE
27      THEN
28          OUTPUT "The value is in the list"
29      ELSE
30          OUTPUT "The value is not found in the list"
31  ENDIF

```

Note:

The pseudocode function

ROUND(Reall : REAL) RETURNS INTEGER

rounds a number to the nearest integer value.

For example: ROUND (4.5) returns 5 and ROUND (4.4) returns 4

- (i) There are four errors in the algorithm.

Write the line of code where an error is present **and** write the correction in **pseudocode**.

Error 1

Correction

Error 2

Correction

Error 3

Correction

Error 4

Correction

[4]

- (ii) A binary search is one algorithm that can be used to search an array.

Identify another searching algorithm.

..... [1]

- (b) The following is an example of a sorting algorithm. It sorts the data in the array `ArrayData`.

```
01 TempValue ← ""
02 REPEAT
03     Sorted ← TRUE
04     FOR Count ← 0 TO 4
05         IF ArrayData[Count] > ArrayData[Count + 1]
06             THEN
07                 TempValue ← ArrayData[Count + 1]
08                 ArrayData[Count + 1] ← ArrayData[Count]
09                 ArrayData[Count] ← TempValue
10                 Sorted ← FALSE
11             ENDIF
12         ENDFOR
13     UNTIL Sorted = TRUE
```

- (i) Complete the trace table for the algorithm given in part (b), for the `ArrayData` values given in the table.

[4]

- (ii) Rewrite lines 4 to 12 of the algorithm in **part (b)** using a WHILE loop instead of a FOR loop.

[3]

- (iii) Identify the algorithm shown in part (b).

[1]

[1]

- (iv) Identify another sorting algorithm.

[1]

BLANK PAGE

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 the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.