Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER			

7142109804

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.

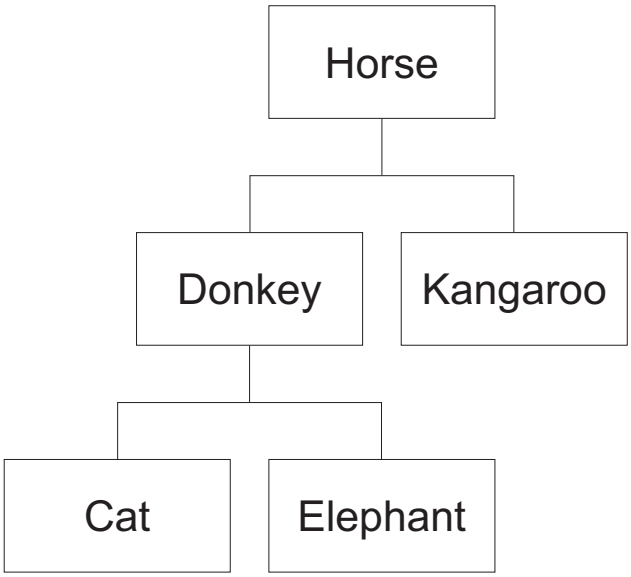
INFORMATION

- 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.

© UCLES 2021

1 An ordered binary tree stores the following data:



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

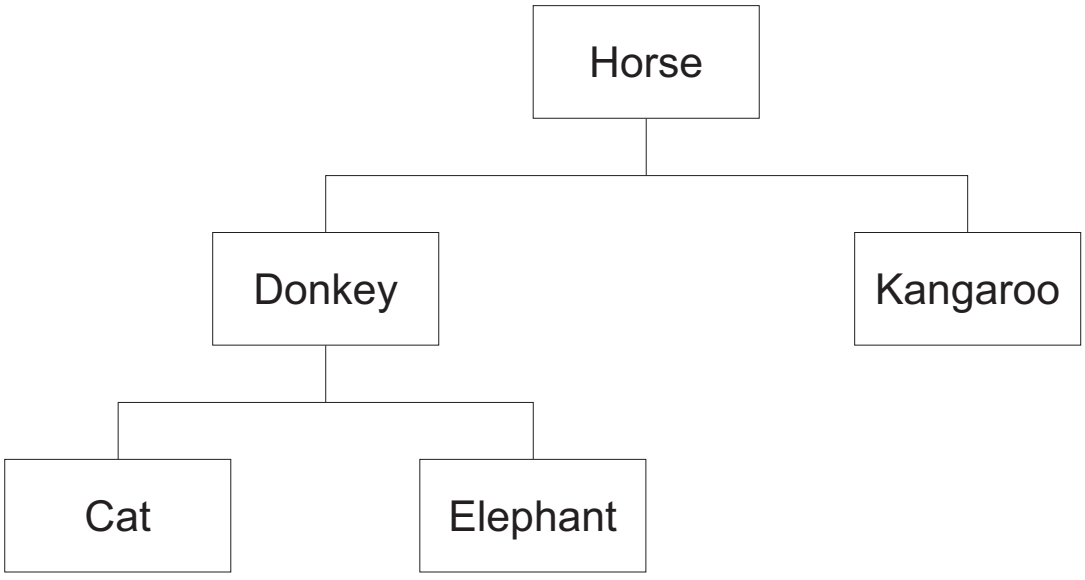
..... [1]

(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]

© UCLES 2021

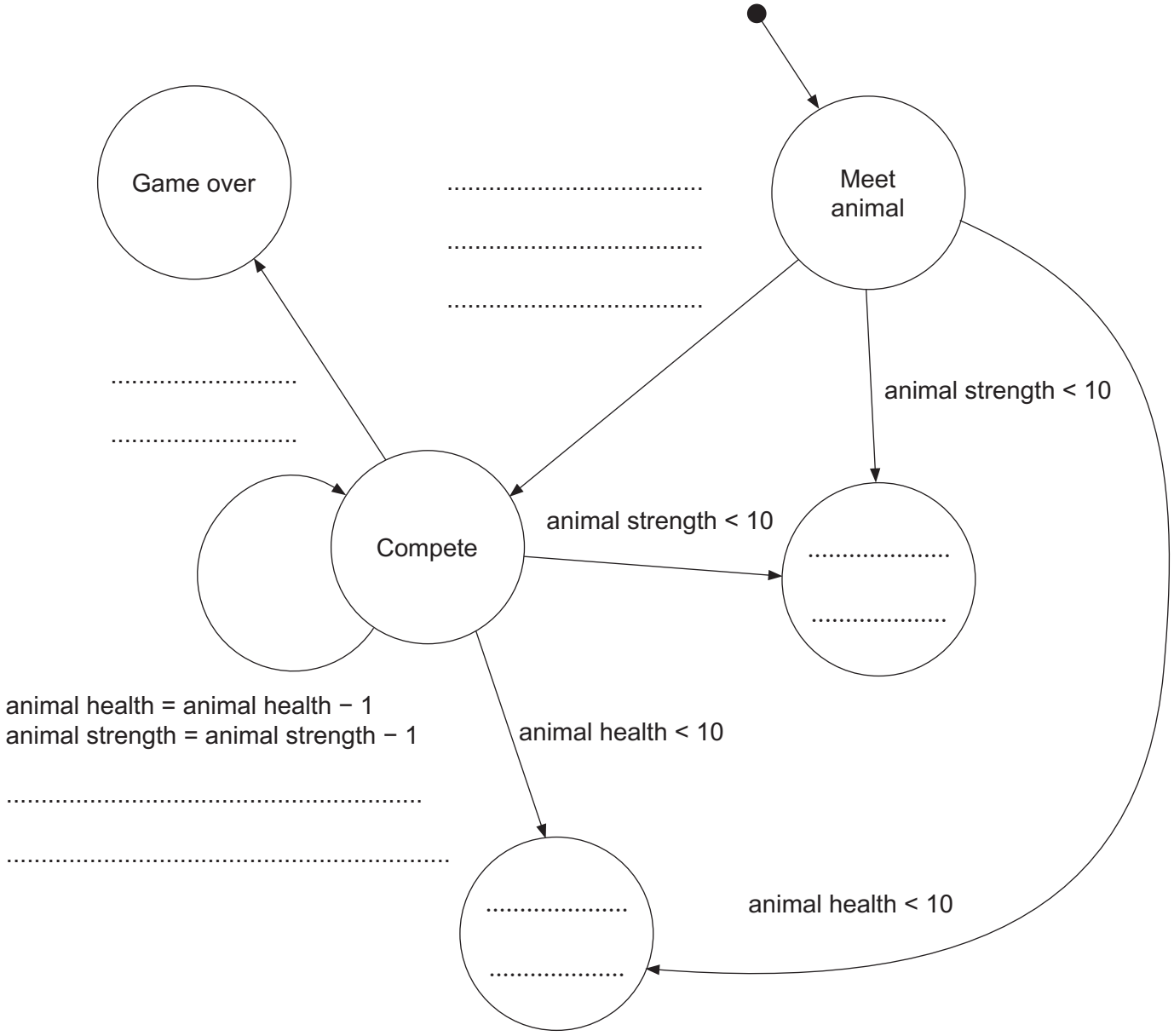
7 Anika is designing a computer game. The user controls a character that moves around a virtual world.
When the character meets an animal:

- 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
	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 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>

© UCLES 2021

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.

© UCLES 2021