OXFORD AQA
INTERNATIONAL QUALIFICATIONS

INTERNATIONAL A-LEVEL
COMPUTER SCIENCE

Unit 3 Advanced Programming

Specimen paper 07:00 GMT Time allowed: 2 hour 30 minutes

Materials
For this paper you must have access to:
- a computer
- a printer
- appropriate software
- the Electronic Answer Document.

Information
- The marks for questions are shown in brackets.
- The maximum mark for this paper is 90.
- You must print and collate this document immediately after the exam time ends.
- The question paper is divided into **two** sections.
- You may use a bilingual dictionary.
- You must **not** use an English dictionary.
- You may use a calculator.

Instructions
- Type the information required on the front of your **Electronic Answer Document**.
- Answer **all** questions.
- Enter your answers for **Section A** into the **Electronic Answer Document**.
- Include the evidence required for your answers to **Section B** in your **Electronic Answer Document**.
- Your programmed solutions to questions must use **C#**, **Python** or **VB.Net**.
- Before the start of the examination make sure your **centre number**, **candidate name** and **candidate number** are shown clearly in the footer of every page (also at the top of the front cover) of your **Electronic Answer Document**.

Advice
You are advised to allocate time to each section as follows:
Section A – 25 minutes; **Section B** – 2 hours 5 minutes.

At the end of the examination
Tie together all your printed **Electronic Answer Document** pages and hand them to the Invigilator.

Warning
It may not be possible to issue a result for this paper if your details are not on every page of your **Electronic Answer Document**.

Section A

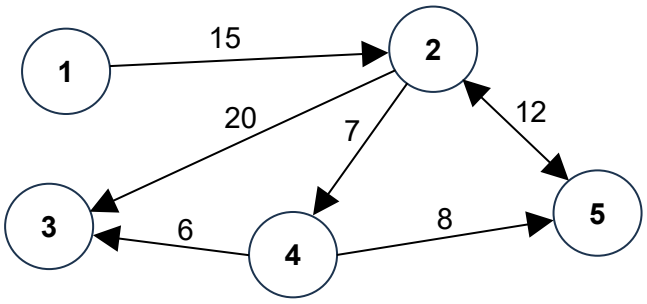
You are advised to spend **25 minutes** on this section which is worth **15 marks**.

Type your answers to **Section A** in your Electronic Answer Document.

You **must save** this document at regular intervals.

0 1 **Figure 1** shows a graph.

Figure 1



0 1 . 1 How many of the following statements about the graph in **Figure 1** are true?

Statements
A. The graph contains cycle(s).
B. The graph is a binary tree.
C. The graph is a tree.
D. The graph is directed.
E. The graph is weighted.

[1 mark]

0 1 . 2 Complete the unshaded cells in **Table 1** to show how the graph in **Figure 1** would be represented as an adjacency matrix.

Each cell **must** contain a number.

Table 1

		To				
		1	**2**	**3**	**4**	**5**
From	**1**					
	2					
	3					
	4					
	5					

Copy the contents of the unshaded cells in **Table 1** into the table in your Electronic Answer Document.

[2 marks]

0 1 . 3 Describe the circumstances when it would be more appropriate to use an adjacency matrix to represent a graph instead of an adjacency list.

[2 marks]

0 2 . 1 Encapsulation is an important principle of object-oriented programming.

Describe what is meant by encapsulation.

[2 marks]

0 2 . 2 Describe the difference between an attribute that has a public access modifier and an attribute that has a protected access modifier.

[2 marks]

0 3 . 1 Data structures can be static or dynamic.

Describe **two** differences between a static data structure and a dynamic data structure.

[2 marks]

0 3 . 2 The pseudocode in **Figure 2** adds an item to a priority queue.

Figure 2

```
IF Queue Is Not Full THEN
  CheckPos ← Rear
  Inserted ← False
  WHILE Inserted = False
    IF ItemToEnqueue.Priority ≤ Queue[CheckPos].Priority
    OR CheckPos = -1 THEN
      Queue[CheckPos + 1] ← ItemToEnqueue
      Inserted ← True
    ELSE
      Queue[CheckPos + 1] ← Queue[CheckPos]
      CheckPos ← CheckPos - 1
    ENDIF
  ENDWHILE
  Rear ← Rear + 1
ENDIF
```

The queue is represented using a static array of records named `Queue`. Each record has two parts:

- the `Value` which represents the data value of the queue item
- the `Priority` which represents the priority of the queue item. The highest priority is 10 and the lowest priority is 1.

Table 2 shows the contents of the `Queue` array at a point in time.

Table 2

	Index					
	[0]	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**
Value	JobA	JobM	JobB	JobT		
Priority	10	10	5	3		

`Rear = 3`

A new item, the `ItemToEnqueue` which has a `Value` of `JobX` and a `Priority` of `7` is to be inserted into the queue.

Complete **Table 3** by writing the contents of the array `Queue` at the end of each iteration of the WHILE loop, when the pseudocode in **Figure 2** is applied to the queue at the top of **Table 3**, to insert the new item with a `Value` of `JobX` and a `Priority` of `7`.

The initial contents of `Queue` has been completed for you.

Table 3

		Index					
		[0]	[1]	[2]	[3]	[4]	[5]
At start	Value	JobA	JobM	JobB	JobT		
	Priority	10	10	5	3		
After 1 iteration	Value						
	Priority						
After 2 iterations	Value						
	Priority						
After 3 iterations	Value						
	Priority						

Copy the contents of the unshaded cells in **Table 3** into the table in your Electronic Answer Document.

[4 marks]

Turn over for the next section

Section B

You are advised to spend **2 hours 5 minutes** on this section which is worth **75 marks**.

Enter your answers in your Electronic Answer Document.

Ensure that the code and test evidence entered into your Electronic Answer Document are big enough to allow the examiner to read them.

You do **not** need to use object-oriented programming techniques in your solutions unless a question asks you to do so.

You **must save** this document and your programs at regular intervals.

0 4 The rail fence cipher encrypts a message by writing the message in a diagonal zig zag along a series of rails and then concatenating together the letters on each rail to make the encrypted ciphertext message.

The number of rails used to encrypt a message can be varied and is the cipher's key.

Figure 3 shows the message OXFORDAQA being encrypted using the rail fence cipher with a key of 3, which means that 3 rails are used in the encryption process:

Figure 3

Plaintext: OXFORDAQA Key: 3

Rail 1	O				R				A
Rail 2		X		O		D		Q	
Rail 3			F				A		

Ciphertext: ORAXODQFA

Figure 4 shows the message OXFORDAQA being encrypted using the rail fence cipher with a key of 4, which means that 4 rails are used in the encryption process:

Figure 4

Plaintext: OXFORDAQA Key: 4

Rail 1	O						A		
Rail 2		X				D		Q	
Rail 3			F		R				A
Rail 4				O					

Ciphertext: OAXDQFRAO

Task

Write a program that encrypts a message using the rail fence cipher.

The program should display a prompt asking the user to enter the message to encrypt and the key to use and should then display the encrypted message.

Test

Test that your program works by encrypting the message `COMPUTER` with a key of `3`. Your program should output the ciphertext `CUOPTRME`

Evidence that you need to provide
Include the following in your Electronic Answer Document.

0 4 . 1 Your PROGRAM SOURCE CODE for your entire program.

[12 marks]

0 4 . 2 The output produced when you carry out the test.

This must include the values entered and the output displayed.

[1 mark]

Turn over for the next question

0 5 A binary tree can be represented using three arrays or an array of records.

For example, the binary tree in **Figure 5** can be represented using three arrays, as shown in **Figure 6**.

Figure 5

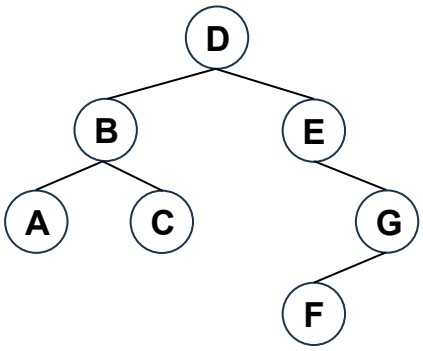


Figure 6

	Three Arrays		
Index	**Label**	**LeftPtr**	**RightPtr**
[0]	D	1	2
[1]	B	3	4
[2]	E	-1	5
[3]	A	-1	-1
[4]	C	-1	-1
[5]	G	6	-1
[6]	F	-1	-1

RootIndex = 0

Figure 7 outlines a recursive method to perform an in-order traversal of a binary tree.

Figure 7

```
SUBROUTINE TraverseTree(CurrentNode)
   IF CurrentNode has LeftSubtree THEN
      TraverseTree(LeftSubtree of CurrentNode)
   ENDIF
   OUTPUT CurrentNode's Label
   IF CurrentNode has RightSubtree THEN
      TraverseTree(RightSubtree of CurrentNode)
   ENDIF
ENDSUBROUTINE
```

Task 1

Write a program that represents the binary tree in **Figure 5** using three arrays or an array of records.

Task 2

Add code to your program so that it performs an in-order traversal of the binary tree.

To achieve full marks for this task, your program must use the **recursive method** outlined in **Figure 7** to perform the traversal.

Test

Test your program works by showing the program's output when an in-order traversal is performed on the binary tree in **Figure 5**.

Evidence that you need to provide
Include the following in your Electronic Answer Document.

0 5 . 1 Your PROGRAM SOURCE CODE for your entire program.

[12 marks]

0 5 . 2 The output produced when you carry out the test.

[1 mark]

Turn over for the next question

0 6 Data about students at a college is to be stored using a hash table.

For each student, the following details will be stored:

- `StudentID` which consists of two uppercase letters followed by three digits, for example CT398
- `Surname`
- `Forename`.

The hash table will contain 100 rows.

The following hash function will be used on the `StudentID`:

$$\text{row} = (\text{position in alphabet of first letter} \times 10 + \text{value of last two digits}) \text{ MOD } 100$$

For example, for the `StudentID` CT398:

$$\begin{aligned} \text{row} &= (\text{position in alphabet of 'C'} \times 10 + 98) \text{ MOD } 100 \\ &= (3 \times 10 + 98) \text{ MOD } 100 \\ &= (30 + 98) \text{ MOD } 100 \\ &= 128 \text{ MOD } 100 \\ &= 28 \end{aligned}$$

Task 1

Using **object-oriented programming**, create a hash table class that can be used to store the details of students. Your hash table class should:

- Create a hash table with 100 rows.
- Have a method `AddStudent` which:
  - takes a student's `StudentID`, `Surname` and `Forename` as parameters
  - stores the student's details in the correct row in the hash table
  - checks if a collision would occur when a student is added and if so, displays a suitable error message instead of adding the student's details.
- Have a method `LookUpStudent` which:
  - takes a student's `StudentID` as a parameter
  - displays the student's `Surname` and `Forename` if a student with the given `StudentID` exists in the hash table
  - displays a suitable error message if no such student exists.
- Have a method `DeleteStudent` which:
  - takes a student's `StudentID` as a parameter
  - removes the student's details from hash table if a student with the given `StudentID` exists in the hash table.

You will be able to achieve some, but not all of the marks for this task if you do not use object-oriented programming techniques.

If you are working in **Python**, you should follow the convention of using two underscores __ to represent a private attribute.

Test

Test that your hash table class works by writing code that carries out the following tasks in the order given:

- adds student Peter Smith with `StudentID` DH409 to the hash table
- adds student Ibrahim Saleem with `StudentID` EP708 to the hash table
- looks up the student with `StudentID` DH409
- deletes the student with `StudentID` DH409
- looks up the student with `StudentID` DH409

Evidence that you need to provide
Include the following in your Electronic Answer Document.

0 6 . 1 Your PROGRAM SOURCE CODE for your entire program.

[21 marks]

0 6 . 2 The output produced when you carry out the **five** tests.

[1 mark]

Turn over for the next question

0 7

The game three-in-a-row is played on a vertical board with six columns and five rows.

Two players drop counters into the columns of the board in turn. A dropped counter falls either to the bottom of the column or rests on any counters already dropped in the column. The first player's counters have a 1 written on them and the second player's counters have a 2 written on them.

The first player to get a line of three of their counters in a row wins the game. A winning line can be in a **horizontal** or **vertical** direction. A line of three counters on a diagonal does not result in a win.

Figure 6 shows a game that has just been won by player one.

- Each player's counter is represented by their player number.
- The row and column numbers are shown around the board.
- The winning line is a vertical line in column 4.

Figure 6

	Col 0	Col 1	Col 2	Col 3	Col 4	Col 5	
Row 4							TOP
Row 3							
Row 2				2	**1**		
Row 1			2	2	**1**		
Row 0	1	2	2	1	**1**	1	BOTTOM

Task

Write a program that allows two players to play a game of three-in-a-row. Your program should:

- Use a board with six columns and five rows.
- Display the board before each turn. You do not need to display row or column headings, just the counters.
- Show player one's counters as a 1 and player two's counters as a 2
- Let the players take turns to choose a column to drop a counter into, by entering the column number.
- After a player has entered a column number:
  - check that the column number exists on the board
  - check if the column is already full.
- If the column number does not exist or the column is already full a suitable error message should be displayed and the player should be made to choose another column.
- If the column number exists and the column is not full then the player's counter should be dropped into the column.
- After each turn, a check should be made to see if the player has won and a message should be displayed if they have.
- The game should end when a player has won or every column is full.
- When the game ends, the final board should be displayed.

Your program should make appropriate use of subroutines, parameters and local variables.

Test

Test that your program works by playing a game until a player wins.

Evidence that you need to provide
Include the following in your Electronic Answer Document.

0 7 . 1 Your PROGRAM SOURCE CODE for your entire program.

[26 marks]

0 7 . 2 The output produced when you carry out the test.

This must include the values entered and the output displayed.

[1 mark]

END OF QUESTIONS