A-level
COMPUTER SCIENCE

Paper 1

Wednesday 11 June 2025 Morning Time allowed: 2 hours 30 minutes

Materials

For this paper you must have:

- a computer
- a printer
- appropriate software
- the Electronic Answer Document
- an electronic version and a hard copy of the Skeleton Program
- an electronic version and a hard copy of the Preliminary Material.

You must **not** use a calculator.

Instructions

- Type the information required on the front of your Electronic Answer Document.
- 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.
- Enter your answers into the Electronic Answer Document.
- Answer **all** questions.
- Save your work at regular intervals.

Information

- The marks for questions are shown in brackets.
- The maximum mark for this paper is 100.
- No extra time is allowed for printing and collating.
- The question paper is divided into **four** sections.

Advice

You are advised to allocate time to each section as follows:
Section A – 40 minutes; **Section B** – 20 minutes; **Section C** – 20 minutes; **Section D** – 70 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 no longer than **40 minutes** on this section.

Enter your answers for **Section A** in your Electronic Answer Document.

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

0 1 Two data structures that can be used to store a collection of data records are a hash table and a sorted list.

0 1 . 1 Explain why a record in a hash table can normally be found more quickly than a record in a sorted list.

[1 mark]

0 1 . 2 Describe the steps involved in adding a new record to a hash table.

[4 marks]

0 1 . 3 Explain why the time taken to find a specific record in a hash table can increase when lots of records are stored in the table.

[1 mark]

Linear search and binary search are two algorithms that could be used to find a record in a sorted list.

0 1 . 4 Explain why the linear search algorithm has time complexity of O(n).

[1 mark]

0 1 . 5 Explain why the binary search algorithm has time complexity of O(log n).

[1 mark]

0 1 . 6 State **one** advantage of a linear search compared to a binary search.

[1 mark]

If the list of records was unsorted, a bubble sort algorithm or a merge sort algorithm could be used to sort the records.

0 1 . 7 Explain why both these sorting algorithms are tractable.

[1 mark]

0 1 . 8 After how many passes is bubble sort guaranteed to have sorted any list of 100 items?

[1 mark]

0 2 Define the term decomposition.

[2 marks]

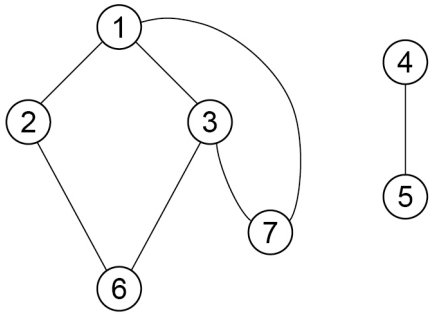
Dijkstra's algorithm is an example of a breadth-first traversal algorithm.

0 3 . 1 State the purpose of Dijkstra's algorithm.

[1 mark]

Figure 1 shows a graph containing seven nodes.

Figure 1



0 3 . 2 State **both** the reasons why the graph in **Figure 1** is not a tree.

[2 marks]

0 3 . 3 Complete the unshaded cells in **Table 1** to show how the graph in **Figure 1** could be represented as an adjacency matrix.

Table 1

	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							

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

[2 marks]

Turn over ►

Figure 2 shows the graph in **Figure 1** represented as an adjacency list, called `AL`. An adjacency list is a list of lists.

Figure 2

Node	Index 1	2	3
1	2	3	7
2	1	6	
3	1	6	7
4	5		
5	4		
6	2	3	
7	1	3	

`AL[1][3]` is equal to 7 as it refers to the 3rd position in the list for node 1

`LENGTH(AL[6])` is equal to 2 as there are two items in the list for node 6

Figure 3 contains pseudo-code for an algorithm that finds out if there is a path between two nodes in the graph represented by the adjacency list `AL`. Both subroutines in the algorithm make use of an array `V`.

Figure 3

```
SUBROUTINE IsPath(start, end)
  FOR j = 1 TO LENGTH(AL)
    V[j] = 0
  END FOR
  FOR j = 1 TO LENGTH(AL)
    IF V[j] = 0 THEN
      IF Traverse(start, end) = True THEN
        RETURN True
      END IF
    END IF
  END FOR
  RETURN False
END SUBROUTINE


SUBROUTINE Traverse(start, end)
  IF start = end THEN
    RETURN True
  END IF
  V[start] = 1
  FOR i = 1 TO LENGTH(AL[start])
    IF V[AL[start][i]] = 0 THEN
      IF Traverse(AL[start][i], end) = True THEN
        RETURN True
      END IF
    END IF
  END FOR
  RETURN False
END SUBROUTINE
```

0 3 . 4 Describe **one** of the base cases for the recursive subroutine `Traverse`.

[1 mark]

Figure 4 shows a trace table for the subroutine call `Traverse(1, 6)`

The subroutine call will return a value of `True` as there is a path from node 1 to node 6 in the graph shown in **Figure 1**, on page 3.

Figure 4

		V						
Subroutine call to `Traverse`	**`i`**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
		0	0	0	0	0	0	0
`Traverse(1, 6)`		1						
	1							
`Traverse(2, 6)`			1					
	1							
	2							
`Traverse(6, 6)`								

Question 3 continues on the next page

Figures 2 and **3** are repeated here to help you answer Question **03.5**.

Figure 2

Node	Index		
	1	**2**	**3**
1	2	3	7
2	1	6	
3	1	6	7
4	5		
5	4		
6	2	3	
7	1	3	

Figure 3

```
SUBROUTINE IsPath(start, end)
  FOR j = 1 TO LENGTH(AL)
    V[j] = 0
  END FOR
  FOR j = 1 TO LENGTH(AL)
    IF V[j] = 0 THEN
      IF Traverse(start, end) = True THEN
        RETURN True
      END IF
    END IF
  END FOR
  RETURN False
END SUBROUTINE


SUBROUTINE Traverse(start, end)
  IF start = end THEN
    RETURN True
  END IF
  V[start] = 1
  FOR i = 1 TO LENGTH(AL[start])
    IF V[AL[start][i]] = 0 THEN
      IF Traverse(AL[start][i], end) = True THEN
        RETURN True
      END IF
    END IF
  END FOR
  RETURN False
END SUBROUTINE
```

0 3 . 5 Complete the unshaded cells in **Table 2** to show the result of tracing the algorithm shown in **Figure 3** using the adjacency list `AL` in **Figure 2** for the subroutine call `IsPath(3, 7)`

Table 2

			V						
j	**Subroutine call to `Traverse`**	**i**	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Copy the contents of the unshaded cells in **Table 2** into the table in your Electronic Answer Document.

[6 marks]

Turn over ►

Section B

You are advised to spend no more than **20 minutes** on this section.

Enter your answers to **Section B** in your Electronic Answer Document.

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

The question in this section asks you to write program code **starting from a new program/project/file**.

You are advised to **save** your program at regular intervals.

0 4 Write a program that uses a transposition cipher to encrypt a string entered by the user.

After the user enters the string to encrypt, they enter the number of columns to use. Different numbers of columns result in different encrypted versions of the same string.

Any non-alphabetic characters in the string entered by the user are ignored and will not appear in the encrypted text.

To encrypt the string, the characters from the user's input are written into a grid (left to right, top to bottom) of cells which has the number of columns specified by the user. The characters are then read from the grid in a different order (top to bottom, left to right).

Example 1

If the user enters the string `hypothetically` and `3` for the number of columns, then the encrypted version of that string will be `hoeclyttayphil`

h	y	p
o	t	h
e	t	i
c	a	l
l	y	

Example 2

If the user enters the string `hypothetically` and `5` for the number of columns, then the encrypted version of that string will be `hhayelptloiytc`

h	y	p	o	t
h	e	t	i	c
a	l	l	y	

Example 3

If the user enters the string `How many, how rarely?` and `4` for the number of columns, then the encrypted version of that string will be `Haoronwewyrlmhay`

H	o	w	m
a	n	y	h
o	w	r	a
r	e	l	y

The spaces and punctuation characters have been removed in the encrypted string.

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

0 4 . 1 Your PROGRAM SOURCE CODE.

[12 marks]

0 4 . 2 SCREEN CAPTURE(S) showing the results of testing the program by entering the string:

```
Hello there, are you ok?
```

and `4` for the number of columns.

[1 mark]

Turn over for the next section

There are no questions printed on this page

Section C

You are advised to spend no more than **20 minutes** on this section.

Enter your answers to **Section C** in your Electronic Answer Document.

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

These questions refer to the **Preliminary Material** and the **Skeleton Program**, but **do not** require any additional programming.

Refer **either** to the **Preliminary Material** issued with this question paper **or** your electronic copy.

0 5 This question is about the `ConvertToRPN` subroutine.

0 5 . 1 Describe **two** advantages of using Reverse Polish Notation (RPN) instead of infix notation to represent expressions.

[2 marks]

0 5 . 2 A dictionary is used to store the precedence of the allowed operators.

Explain the concept of a dictionary.

[2 marks]

0 5 . 3 State the type of data structure that has been represented using the `Operators` list.

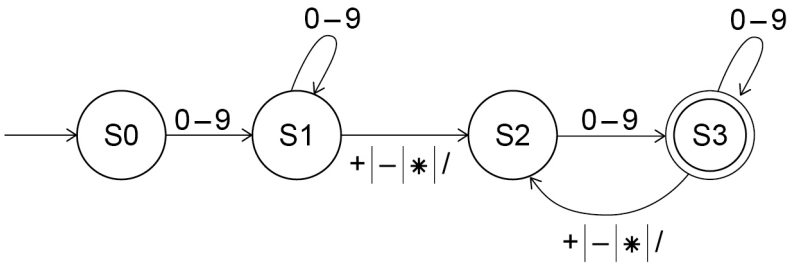
[1 mark]

Turn over for the next question

0 6 This question is about the `CheckIfUserInputValid` subroutine.

Figure 5 shows a finite state machine (FSM) represented as a state transition diagram. The FSM is equivalent to the regular expression in the `CheckIfUserInputValid` subroutine.

Figure 5



0 6 . 1 Complete the unshaded cells in **Table 3** to represent the FSM in **Figure 5** as a state transition table.

Part of the first row has been completed for you.

Table 3

Initial state	Input(s)	New state
	0–9	

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

[2 marks]

Figure 6 and **Figure 7** show two attempts to represent the regular expression in the `CheckIfUserInputValid` subroutine using a state transition diagram. There are errors in both diagrams, as the FSM will either accept some strings that do not match the regular expression or not accept some strings that do match the regular expression.

Figure 6

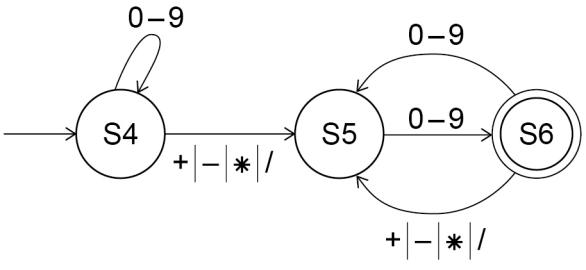
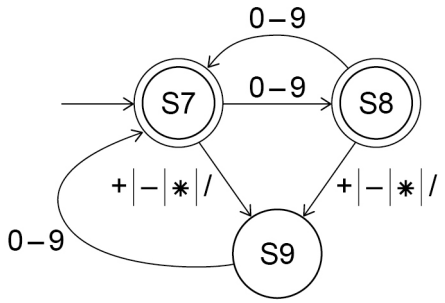


Figure 7



0 6 . 2 Explain which strings accepted by the FSM in **Figure 5** are **not** accepted by the FSM in **Figure 6**.

[2 marks]

0 6 . 3 Explain which strings accepted by the FSM in **Figure 7** are **not** accepted by the FSM in **Figure 5**.

[2 marks]

0 6 . 4 Explain the purpose of the metacharacter `$` in the regular expression in the `CheckValidNumber` subroutine.

[1 mark]

0 6 . 5 Explain the difference between the metacharacters `*` and `?` in regular expressions.

[1 mark]

0 6 . 6 A regular expression can be used to describe a set.

State **two** properties of sets that do not have to be true for a list.

[2 marks]

Turn over ►

Infix expressions can normally use brackets.

The infix expressions entered by the user in the Skeleton Program cannot use brackets.

0 6 . 7 Explain why a regular expression could not be used to check if an infix expression entered by the user was valid if the infix expression was allowed to contain brackets.

[1 mark]

Figure 8 contains three rules for a language represented using Backus-Naur Form (BNF).

Figure 8

```
<operator> ::= +|-|*|/
<digit>    ::= 0|1|2|3|4|5|6|7|8|9
<number>   ::= <digit>|<digit><number>
```

0 6 . 8 Add BNF rule(s) to the language shown in **Figure 8** for an `expression`.

An `expression` should match all infix expressions currently accepted by the Skeleton Program and also allow the infix expression to use brackets.

[3 marks]

0 7 This question is about the `EvaluateRPN` subroutine.

0 7 . 1 Describe an infix expression that:

- would cause the subroutine `CheckIfUserInputValid` to return `True` and
- could be successfully converted to RPN by the `ConvertToRPN` subroutine

but that would **not** be evaluated correctly by the `EvaluateRPN` subroutine.

[1 mark]

0 7 . 2 Explain the purpose of the final selection structure in the `EvaluateRPN` subroutine.

Your explanation should include a description of what the possible values returned by the subroutine represent.

[2 marks]

0 8 State the identifier for a variable in the Skeleton Program that could represent a vector.

[1 mark]

Turn over for the next section

Section D

You are advised to spend no more than **70 minutes** on this section.

Enter your answers to **Section D** in your Electronic Answer Document.

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

These questions require you to load the **Skeleton Program** and to make programming changes to it.

0 9 This question refers to the `PlayGame` subroutine.

The Skeleton Program is to be changed so that there are penalty positions in the list of targets.

If there is a number shown in the second position of the list of targets, then the player's score is to be reduced by five.

If there is a number shown in the first position of the list of targets, then the player's score is to be reduced by the value of the number that is ten larger than the number shown in that position.

If there are numbers in the first and second positions of the list of targets, then the player's score will be reduced by five and by the value of the number that is ten larger than the number shown in the first position of the list of targets.

The program should check if there are numbers shown in the penalty positions after reducing the player's score by one.

Example

		3	80	17				62	7	31	11	0	18	19	22	41	13	22	75

Numbers available: 2 7 8 3 4

Current score: 4

Player enters the expression 2*7

This expression evaluates to 14. This is not one of the targets, so the player's score will reduce by one.

	3	80	17				62	7	31	11	0	18	19	22	41	13	22	75	1	

Numbers available: 2 7 8 3 4

Current score: 3

Player enters the expression 2*7

This expression evaluates to 14. This is not one of the targets, so the player's score will reduce by six (1 + 5).

3	80	17				62	7	31	11	0	18	19	22	41	13	22	75	1	33

Numbers available: 2 7 8 3 4

Current score: –3

Player enters the expression 2*7

This expression evaluates to 14. This is not one of the targets so the player's score will reduce by 19 (1 + 5 + 10 + 3).

The game now ends and the final score (–22) will be displayed.

What you need to do

Task 1
Modify the subroutine `PlayGame` so it reduces the player's score if there are numbers shown in a penalty position.

Task 2
Test that the changes you have made work:

- run the Skeleton Program
- select the training game
- enter the expression `4*5` six times.

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

0 9 . 1 Your PROGRAM SOURCE CODE for the amended subroutine `PlayGame`.

[4 marks]

0 9 . 2 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

1 0 This question extends the Skeleton Program by allowing the player to use exponentiation in the expressions they enter.

The player will be allowed to use the `^` symbol to denote exponentiation in an expression they enter. Exponentiation has higher precedence than the other operators.

The four operators currently allowed in expressions are left-associative.

Example

`12-4-3` is equal to `5`
as `12-4` is evaluated to 8 and then `8-3` is evaluated to 5

If subtraction was right-associative, then `12-4-3` would evaluate to `11`
as `4-3=1` and `12-1=11`

The exponentiation operator will be right-associative.

Example

`2^3^2` is equal to `512`
as `3^2` is evaluated to 9 and then `2^9` is evaluated to 512

If exponentiation was left-associative, then `2^3^2` would evaluate to `64`
as `2^3=8` and `8^2=64`

What you need to do

Task 1
Modify the `CheckIfUserInputValid`, `ConvertToRPN` and `EvaluateRPN` subroutines so the exponentiation operator is allowed and works correctly.

Note: you will be able to achieve most of the marks for this question if you implement the exponentiation operator as left-associative.

Task 2
Test that the changes you have made work:

- run the Skeleton Program
- select the training game
- enter `512/2^3^2+8`
- enter `3^2`

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

1 0 . 1 Your PROGRAM SOURCE CODE for the amended subroutines `CheckIfUserInputValid`, `EvaluateRPN`, and `ConvertToRPN`.

[7 marks]

1 0 . 2 SCREEN CAPTURE(S) showing the requested test.

[1 mark]

1 1 This question extends the Skeleton Program by telling the player all the values that appear more than once in the list of targets.

Before displaying the current score, the program should display an additional list containing all the values that appear more than once in the list of targets. In this additional list, each value should only be displayed once.

What you need to do

Task 1
Create a new subroutine called `DisplayTargetMultiples` that will display all the values that appear more than once in the targets shown to the player.

Task 2
Modify the `DisplayState` subroutine so that it calls the subroutine `DisplayTargetMultiples` immediately before the current score is displayed.

Task 3
Test that the changes you have made work:

- run the Skeleton Program
- select the training game.

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

1 1 . 1 Your PROGRAM SOURCE CODE for the new subroutine `DisplayTargetMultiples` and the amended subroutine `DisplayState`.

[11 marks]

1 1 . 2 SCREEN CAPTURE(S) showing the requested test.

[1 mark]

Turn over for the next question

1 2 This question extends the Skeleton Program so that the program will tell the player if they can make one of the targets using an expression that contains exactly three of the values from the numbers available.

Your program does **not** need to work with the exponentiation operator from **Question 10**.

What you need to do

Task 1
Create a new subroutine called `CanGetATargetUsingThreeNumbers` that will return `True` if it is possible to create an expression using exactly three of the values in `NumbersAllowed` that matches any of the targets shown to the player.

Otherwise, it should return `False`.

Task 2
Modify the `DisplayState` subroutine so that it calls the subroutine `CanGetATargetUsingThreeNumbers` and then displays an appropriate message saying if it is possible or not to make one of the targets using exactly three of the values from the available numbers.

Task 3
Test that the changes you have made work:

- run the Skeleton Program
- select the training game
- enter `3+8-2`

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

1 2 . 1 Your PROGRAM SOURCE CODE for the new subroutine `CanGetATargetUsingThreeNumbers` and the amended `DisplayState` subroutine.

[11 marks]

1 2 . 2 SCREEN CAPTURE(S) showing the requested test.

[1 mark]

1 2 . 3 State how many valid expressions there are that use exactly three numbers.

You should assume that all the numbers available are different and that the exponentiation operator from **Question 10** has **not** been used.

You should show your working.

[2 marks]

END OF QUESTIONS

There are no questions printed on this page

There are no questions printed on this page

There are no questions printed on this page

There are no questions printed on this page