NAME DATE STARTED TARGET END DATE

GCE A LEVEL — COMPUTER SCIENCE UNIT 3 QUESTION PACK

1500U30-1 · 2015 spec Unit 3 Topic 1 · A2 unit, first sat 2017, 100 marks, 2h paper

REVISE.wales

COMPUTER SCIENCE — UNIT 3 · Advanced Data Structures — Linked Lists, Stacks, Queues, Trees, Graphs & Hashing

Topic 3.1 — Static and dynamic structures: arrays, stacks, queues, linked lists, trees, hash tables and graph representations

Operations on stacks (LIFO) and queues (FIFO), inserting/deleting items in singly linked lists (with next-pointer tables), constructing and traversing binary trees, populating hash tables and resolving collisions, and representing weighted graphs as 2-D arrays for shortest-path problems.

2015 specification · current

Estimated time for entire question pack: ~1 h 54 min

Derived from the Unit 3 pace of ~1.5 min/mark, padded for written-prose answers (76 marks over 9 questions).

*You are advised to **not** attempt to complete all of this in one sitting.*

ABOUT THIS QUESTION PACK

This is a **comprehensive topic question pack**, not a single mock paper. It contains every question from the WJEC A2 Unit 3 papers (Summer 2017 – Summer 2024, COVID gap) that maps onto Topic 3.1 of the 2015 specification.

Questions are ordered by source paper date.

INSTRUCTIONS

Use black ink or black ball-point pen. Show all working. A calculator is allowed where useful.

For Examiner's use only

Q	Source	Max	Mark
1	S17 Q11	14	
2	S18 Q1	8	
3	S19 Q1	8	
4	S19 Q6	8	
5	S22 Q1	8	
6	S23 Q1	8	
7	S23 Q7	10	
8	S24 Q1	8	
9	S24 Q12	4	
Total		**76**	

Advanced Data Structures — Linked Lists, Stacks, Queues, Trees, Graphs & Hashing — what the spec asks

WJEC GCE A Level Computer Science (from 2015) · Unit 3: Programming & System Development · Topic 3.1.

Stacks & queues

- Stack: LIFO — `push` / `pop` at the top; used in call stacks, undo, expression evaluation.
- Queue: FIFO — `enqueue` at rear, `dequeue` at front; used in print spoolers, BFS.
- Implement using array + pointers (top, front, rear) or linked list.
- Detect underflow (empty) and overflow (full, for fixed-size).

Linked lists

- Each node holds data + next pointer; head pointer marks the start.
- Insert: update predecessor's next pointer; cheap O(1) once position is known.
- Delete: bypass node by updating predecessor's pointer.
- Ordered list: traverse to find correct insert point; O(n) search.

Binary trees

- Each node has a key + left and right child pointers.
- BST property: left subtree < node < right subtree.
- Traversals: pre-order (root, L, R) — copy tree; in-order — sorted output; post-order — delete.
- Balanced tree gives O(log n) search; degenerate (linked-list) gives O(n).

Hash tables

- Hash function maps key → index in a fixed-size array.
- Search / insert in O(1) average regardless of size, provided low load factor.
- Collision handling: open addressing (probing) or chaining (linked list per slot).
- Choose hash function to spread keys evenly; resize when load factor too high.

Graphs & shortest paths

- Vertices + edges; edges may carry weights (e.g. distance, cost).
- Adjacency matrix: 2-D array, cell [i][j] = edge weight, 0/∞ if no edge.
- Shortest-path algorithm finds min-cost route between two vertices.
- Don't forget node-traversal cost if specified in the question.

Arrays — 2-D and 3-D

- `myArray[row][col]` indexes 2-D; `myArray[i][j][k]` indexes 3-D.
- Update single element: assign directly to its index.
- Often used to model grids, calendars, image data, networks.
- Memory grows as $O(n^d)$ where d is number of dimensions.

Crib sheet »

Advanced Data Structures — Linked Lists, Stacks, Queues, Trees, Graphs & Hashing in one page

Quick-reference notes — revisit before each question.

Stack ops

push(x): top ← top + 1; arr[top] ← x.
pop(): x ← arr[top]; top ← top − 1; return x.
Used for: call stack, undo, expression evaluation, DFS.

Queue ops

enqueue(x): rear ← rear + 1; arr[rear] ← x.
dequeue(): x ← arr[front]; front ← front + 1; return x.
Used for: print spool, BFS, ticket lines.

Linked-list insert

Find predecessor P.
new.next ← P.next.
P.next ← new.
For an ordered list, traverse until next.data > new.data.

Tree traversals

Pre-order: **Root**, L, R — copy tree.
In-order: L, **Root**, R — sorted output for BST.
Post-order: L, R, **Root** — safely delete tree.

Hash table

index = hash(key) MOD size.
O(1) average insert/search.
Collisions: chaining (linked list per slot) or probing.
Resize when load factor > ~0.7.

Graph as 2-D array

adj[i][j] = weight of edge i→j, 0 if no edge.
Symmetric for undirected graphs.
Diagonal = 0 (no self-loops).
Add per-node traversal cost only if question states it.

10. A software company carries out a design review as part of its quality control procedures.

Identify **three** tasks that would be carried out during a design review. [3]

11. A linked list can be ordered or unordered.

(a) Explain the difference between searching for an item in an ordered list compared with searching an unordered list. [2]

(b) The table below includes an unordered list of names.

Index	Data	Next Pointer (1)	Next Pointer (2)	Next Pointer (3)
0	Smith			
1	Jones			
2	Ahmed			
3	Lewis			
4	Thomas			
5	Brown			
6				
7				
8				
9				

(i) Copy the table and complete the **Next Pointer (1)** column to link the list in ascending alphabetical order. [3]

(ii) Add Murphy and Collins to the linked list and complete the **Next Pointer (2)** column. [4]

(iii) Complete the **Next Pointer (3)** column to delete Smith. [2]

(c) Draw a representation of a binary tree using the data items from question **11***(b)* as key values. [3]

12. A debugging tool of an Integrated Development Environment (IDE) enables stepping, break points and variable watches.

Describe the use of stepping, break points and variable watches in the debugging of programs. [6]

13. Compare the Waterfall and Agile approaches to systems analysis and discuss suitable programming paradigms for each approach. [10]

END OF PAPER

Answer **all** *questions.*

1. A binary tree structure is designed to contain strings and uses the following rules:

- The left pointer indicates the condition “earlier or at the same position in the alphabet”
- The right pointer indicates the condition “later in the alphabet”

(a) Construct a binary tree using these rules and the data entered in the following order:
Goat, Duck, Fox, Bear, Ant, Cat, Leopard, Owl, Mayfly, Insect, Jaguar, Emu.

You may use the initial letters if you wish. [2]

(b) Carry out a pre-order traversal of the tree and give **one** use of pre-order traversals. [2]

(c) Carry out an in-order traversal of the tree and give **one** use of in-order traversals. [2]

(d) Carry out a post-order traversal of the tree and give **one** use of post-order traversals. [2]

2. The evaluation of a computer based solution should consider system functionality and system performance.

(a) Identify a criterion for the evaluation of the functionality of a system and a criterion for the evaluation of the performance of a system. [2]

(b) Developments in human-computer interaction include natural, immersive and experiential interfaces.

Describe, giving examples, the main characteristics of natural and immersive human computer interfaces. [6]

3. Use a truth table to prove De Morgan’s Law $\overline{A + B} = \overline{A} \cdot \overline{B}$. [3]

4. *(a)* Simplify the following using De Morgan’s Law’s and Boolean identities.
Identify which law or identity you are using: [3]

$$\overline{A.B} + A$$

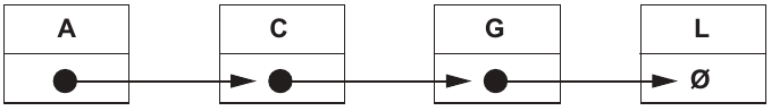
(b) Simplify the following expression using Boolean identities and rules: [5]

$$A.B.(\overline{B} + C) + B.C + B$$

Answer **all** *questions.*

1. *(a)* Explain the differences between stack and queue data structures. [4]

(b) This is a diagram of a linked list in alphabetical order.



(i) Redraw the linked list after the data item 'E' has been added. [2]

(ii) Redraw the amended linked list after the data item 'C' has been deleted. [2]

2. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law.

(a) $A.\overline{A} + A.B + A.\overline{B} + B.\overline{B}$ [5]

(b) $(\overline{A.B}) + A.C + B$ [5]

3. This is an eight-bit number:

$$01101001_2$$

Include this number in a worked example to demonstrate how masking can be used to determine the state of the most significant bit. [3]

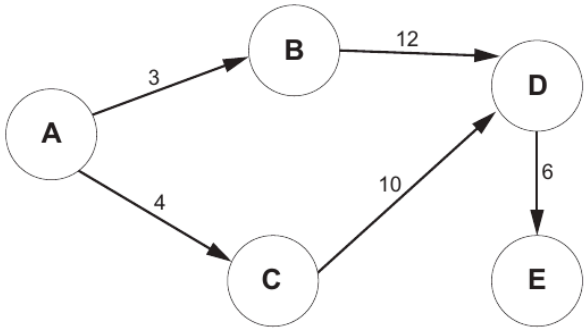
4. *(a)* Describe the term natural language interface. [2]

(b) Using an example, describe a potential use for natural language interfaces. [2]

(c) Describe the potential problems that can be associated with natural language interfaces. [3]

6. *(a)* Explain the purpose of a shortest path algorithm. [4]

(b) This is a diagram of the costs of traversing a network.



The traversal cost for each node is 2.

(i) Show how this network and its traversal costs can be represented using a two dimensional array. [2]

(ii) State the shortest path from node A to E and calculate its cost. [2]

7. *(a)* Explain what is meant by the term programming paradigm. [2]

(b) Describe the difference between procedural and event-driven programming paradigms. [4]

(c) Using examples, describe which programming paradigms would be most suitable when developing different types of software applications. [4]

8. Draw a truth table to prove the following: [4]

A NOR B = NOT A AND NOT B

Answer **all** questions.

1. (a) Explain, giving a suitable example, the operation of a stack data structure. [4]

(b) This is a graphical representation of a two-dimensional array:

alphabet[ ]

	0	1	2	3	4	5	6
0	'a'	'b'	'c'	'd'	'e'	'f'	'g'
1	'h'	'i'	'j'	'k'	'l'	'm'	'n'
2	'o'	'p'	'q'	'r'	's'	't'	'u'
3	'v'	'w'	'x'	'y'	'z'	' '	'.'

(i) Describe how you would update the value **'.'** to **'!'** in the alphabet array. [2]

(ii) Demonstrate how a two-dimensional array could be used to store pupil data in school. [2]

2. (a) Define the term algorithm. [2]

(b) Describe one method of defining algorithms. [2]

(c) Explain the benefits and drawbacks of recursive and non-recursive algorithms. [4]

3. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law where appropriate.

(a) $(A + C).(A.C + A.\overline{C}) + A.C$ [6]

(b) $\overline{X.Y}\ .(\overline{X} + Y) + \overline{Z}$ [5]

4. Write a Binary Search algorithm to search for a value in an array of characters and output its position. You may assume that the search value exists in the array. [8]

Answer **all** questions.

1. (a) Explain the operation of a hash table. [4]

(b) This is a diagram of a hash table:

Key	Value
1001	Apple
1002	Berry
1003	Kiwi
1004	Lime
1005	Mango
1006	Pear
1007	Pineapple

(i) Redraw the hash table after "1008, Orange" has been added and "1003, Kiwi" has been deleted. [2]

(ii) Starting from the modified hash table resulting from (b)(i) redraw the hash table after "1006, Peach" has been added. [2]

2. Explain the differences between procedural and object orientated programming paradigms, giving suitable examples for each. [8]

3. Using the following Truth Table express *p* as a Boolean expression:

A	*B*	*C*	*p*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

[4]

4. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law where appropriate.

(a) $A.(1+C)+\overline{B}.(A+B)$ [5]

(b) $X.\left(\overline{Y+Z}\right)+\overline{Z}.X$ [5]

5. Write a quicksort algorithm, to sort an array of integers into ascending order. [8]

6. State the purpose of validation and verification, giving a suitable method for each. [4]

7. An online grocery store uses binary trees. These binary trees can be traversed using a variety of methods.

(a) Describe the following methods of traversal and give an example of why each method would be used in the grocery store.

(i) In-order traversal [3]

(ii) Post-order traversal [3]

(iii) Pre-order traversal [3]

(b) Draw an example of a balanced binary tree for grocery items. [1]

8. Giving suitable examples, describe the types of software tool that have been designed to assist in the following:

(a) Analysis and planning [3]

(b) Software development [3]

(c) Version management [3]

Answer **all** questions.

1. (a) Explain, giving a suitable example for each, the difference between a two-dimensional and a three-dimensional array. [4]

(b) This is a graphical representation of a two-dimensional string array:

`july[n,n]`

	0	1	2	3	4	5	6
0	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	01	02	03	04	05	06	07
2	08	09	10	11	12	13	14
3	15	16	17	18	19	20	21
4	22	23	24	25	26	27	28
5	29	30	31	X	X	X	

Describe how you would update the empty element in the array to `X`. [2]

(c) This is a representation of a two-dimensional integer array containing the values 1 to 9:

```
2D = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Describe how you would change the value 8 to 10 in the `2D` array. [2]

2. Computer languages such as HTML and CSS are standardised.

(a) Describe the need for the standardisation of computer languages. [4]

(b) Explain, giving a suitable example, the potential difficulties involved in agreeing and implementing standards. [4]

3. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law where appropriate.

(a) $(1.X + \overline{Y.Y}).\overline{X} + X.Z$ [6]

(b) $(\bar{A}.A + A.A).1 + B.C.0$ [6]

4. Write an algorithm to search an unsorted one-dimensional array of strings and replace any duplicate values with the string "X" and then output the array. [8]

10. This is a signed 8-bit integer:

$$00001111_2$$

Include this integer in a worked example to demonstrate how masking can be used to determine the sign of the integer. [3]

11. This algorithm merges two one-dimensional arrays of length *n*. Assume the arrays (`myArray1` and `myArray2`) have already been populated with data.

`num(array)` returns the number of elements currently in the array.

```
1    declare i,j as integer
2    declare myArray1[n]
3    declare myArray2[n]
4    declare myArray3[]
5
6    set i = 0
7    set j = 0
8
9    for i = 0 to len(myArray1) - 1
10        set myArray3[num(myArray3)] = myArray1[i]
11   next i
12
13   for j = 0 to len(myArray2) - 1
14        set myArray3[num(myArray3)] = myArray2[j]
15   next j
16
17   output myArray3
```

(a) Evaluate the efficiency of the algorithm and, using Big O notation, determine the growth rate for time performance. [5]

(b) Determine the growth rate of memory space during a single run of the algorithm. [2]

(c) Identify the type of time complexity and draw a graph of the algorithm to illustrate the order of time performance. Graph paper is not required. [2]

12. Explain, giving a suitable example, the shortest path algorithm. [4]

13. Discuss the potential use of natural language interfaces in human-computer communication to address the problem of communicating with computers.

You should draw on your knowledge, skills and understanding from a number of areas across your computer science course when answering this question. [12]

END OF PAPER

END OF QUESTION PACK

9 questions · 76 marks · ~1 h 54 min
Source: WJEC A2 Computer Science Unit 3 (1500U30-1), Summer 2017–2024, COVID gap
Curated for WJEC Computer Science 2015 spec A2 Unit 3 — Topic 1 (3.1)