3.1 Graph traversal Vectors Mark Scheme

Q1.

(a) **All marks AO2 (analyse)**

	1	2	3	4	5	6
1	0	2	5	3	0	8
2	2	0	1	0	0	0
3	5	1	0	0	0	4
4	3	0	0	0	1	0
5	0	0	0	1	0	5
6	8	0	4	0	5	0

Alternative answer

	1	2	3	4	5	6
1	0	2	5	3	0	8
2		0	1	0	0	0
3			0	0	0	4
4				0	1	0
5					0	5
6						0

Alternative answer

	1	2	3	4	5	6
1	0					
2	2	0				
3	5	1	0			
4	3	0	0	0		
5	0	0	0	1	0	
6	8	0	4	0	5	0

Mark as follows:

1 mark 0s in correct places

1 mark all other values correct

I. non-zero symbols used to denote no edge but only for showing no edge going from a node to itself

2

(b) **All marks for AO1 (understanding)**

Adjacency list appropriate when there are few edges between vertices // when graph/matrix is sparse; **NE**. few edges

Adjacency list appropriate when edges rarely changed;

Adjacency list appropriate when presence/absence of specific edges does not need to be tested (frequently);

A. Alternative words which describe edge, eg connection, line, arc

Max 2

2

(c) **Mark is for AO2 (apply)**

It contains a cycle / cycles;

1

(d) **Mark for AO1 (knowledge)**

A graph where each edge has a weight/value associated with it;

1

(e) **All marks AO2 (apply)**

Mark as follows:

I. output column
1 mark first value of `A` is 2
1 mark second value of `A` is 5 and third value is 3
1 mark fourth and subsequent values of `A` are 8, 3, 7, 4, 9 with no more values after this
1 mark `D[2]` is set to 2 and then does not change

				D						P					
U	Q	V	A	1	2	3	4	5	6	1	2	3	4	5	6
-	1,2,3,4,5,6	-	-	20	20	20	20	20	20	-1	-1	-1	-1	-1	-1
				0											
1	2,3,4,5,6	2	2		2						1				
		3	5			5						1			
		4	3				3						1		
		6	8						8						1
2	3,4,5,6	3	3			3						2			
3	4,5,6	6	7						7						3
4	5,6	5	4					4						4	
5	6	6	9												
6	-														

1 mark `D[3]` is set to 5 and then changes to 3 and does not change again

1 mark correct final values for each position of array `P`

				D						P					
U	Q	V	A	1	2	3	4	5	6	1	2	3	4	5	6
-	1,2,3,4,5,6	-	-	20	20	20	20	20	20	-1	-1	-1	-1	-1	-1
				0											
1	2,3,4,5,6	2	2		2						1				
		3	5			5						1			
		4	3				3						1		
		6	8						8						1
2	3,4,5,6	3	3			3						2			
3	4,5,6	6	7						7						3
4	5,6	5	4					4						4	
5	6	6	9												
6	-														

1 mark correct final values for `D[1]`, `D[4]`, `D[5]`, `D[6]`

				D						P					
U	Q	V	A	1	2	3	4	5	6	1	2	3	4	5	6
-	1,2,3,4,5,6	-	-	20	20	20	20	20	20	-1	-1	-1	-1	-1	-1
				0											
1	2,3,4,5,6	2	2		2						1				
		3	5			5						1			
		4	3				3						1		
		6	8						8						1
2	3,4,5,6	3	3			3						2			
3	4,5,6	6	7						7						3
4	5,6	5	4					4						4	
5	6	6	9												
6	-														

Max 6 marks if any errors

7

(f) **Mark is for AO2 (analyse)**

The shortest distance / time between locations/nodes 1 and 6;

NE distance / time between locations/nodes 1 and 6

R. shortest route / path

1

(g) **All marks AO2 (analyse)**

Used to store the previous node/location in the path (to this node);

Allows the path (from node/location 1 to any other node/location) to be recreated // stores the path (from node/location 1 to any other node/location);

Max 1 if not clear that the values represent the shortest path

Alternative answer

Used to store the nodes that should be traversed;

And the order that they should be traversed;

Max 1 if not clear that the values represent the shortest path

2

[16]

Q2.

(a) John, Rachel, Paul;
R. If not in correct order
I. Incorrect spellings of names, as long as the name is comprehensible
I. Quotation marks

1

(b)

Time Complexity	Tick one box
O(n)	
O(log n)	✔
$O(n^2)$	

A. Alternative symbol which clearly indicates just one box e.g. cross, Y, Yes

R. Answers in which more than one row is ticked

1

(c) **1 mark** for Start Index containing the index of the array item storing the root node (John)
1 mark for John, Hannah and Rachel being stored, each with left and right pointer values storing the indices of the correct child nodes
1 mark for Bradley, Jo, Paul and Tina being stored, each with left and right pointers storing appropriate rogue values e.g. -1, 0, NIL, NULL **R.** A dash or blank as the rogue value

Three example solutions are shown below, but the data items can be stored at any positions within the array, as long as the pointers correctly reflect this.

Start Index = 1

Index	Left Pointer	Data	Right Pointer
[1]	2	John	3
[2]	4	Hannah	5
[3]	6	Rachel	7
[4]	–1	Bradley	–1
[5]	–1	Jo	–1
[6]	–1	Paul	–1
[7]	–1	Tina	–1

Start Index = 4

Index	Left Pointer	Data	Right Pointer
[1]	–1	Bradley	−1
[2]	–1	Jo	–1
[3]	1	Hannah	2
[4]	3	John	6
[5]	–1	Paul	–1
[6]	5	Rachel	7
[7]	–1	Tina	–1

Start Index = 1

Index	Left Pointer	Data	Right Pointer
[1]	2	John	5
[2]	3	Hannah	4
[3]	–1	Bradley	–1
[4]	–1	Jo	–1
[5]	6	Rachel	7
[6]	–1	Paul	–1
[7]	–1	Tina	–1

3

(d) **Difference between Static and Dynamic (2 marks):**

Static structures have fixed (maximum) size whereas size of dynamic structures can change // Size of static structure fixed at compile-time whereas size of dynamic structure can change at run-time;
Static structures can waste storage space/memory if the number of data items stored is small relative to the size of the structure whereas dynamic structures only take up the amount of storage space required for the actual data;
Static structures (typically) store data in consecutive memory locations, which dynamic data structures (typically) do not // Dynamic data structures (can) (require memory to) store pointer(s) to the (next items which static structures typically do not need); **MAX 2**
A. Just one side of points, other side is by implication

Heap (1 mark):
Memory allocated/deallocated at run-time/for new items (to dynamic data structure);
(Provides a) pool of free/unused/available memory;
N.E. To store new items

3

(e) (i) Bradley, Hannah, Jo, John, Paul, Rachel, Tina;
R. If not in correct order
I. Incorrect spelling of names, so long as name is comprehensible
I. Quotation marks

1

(ii) (Ascending) Alphabetic order;
A. Alphabetic, it is sorted

1

(f) Graph may contain cycles / loops / circuits (so must keep track of which nodes already visited);
Graph may not be connected (so some nodes may be unreachable);
Graph may be weighted (so a more complex algorithm that accounts for the weights may be required);
N.E. Graphs can be directed

MAX 1

1

[11]

Q3.

(a) **Mark is for AO1 (understanding)**

It contains a cycle / cycles;

1

(b) **All marks AO2 (apply)**

Vertex (in Figure 1	Adjacent vertices
1	2,3
2	1,3,4
3	1,2,5
4	2
5	3

Mark as follows:
1 mark: Three correct rows;
1 mark: All rows correct;
I Order of items within each list / row.

2

(c) **All marks AO1 (understanding)**

Adjacency list appropriate when there are few edges between vertices / / when graph / matrix is sparse;
when edges rarely changed;
when presence / absence of specific edges does not need to be tested (frequently);
Max 2
A Alternative words which describe edge, eg connection, line.

2

(d) **All marks AO2 (apply)**

				Cat				
NoOfCats	**A**	**B**	**C**	**1**	**2**	**3**	**4**	**5**
5				1				
	2	1	1					
		1	2					
		2			2			
	3	1	1					
		1	2					
		2						
		1	3					
		2						
		3				3		
	4	1	1					
		2						
		3						
		4					1	
	5	1	1					
		2						
		3						
		4						
		5						1

Mark as follows:
1 mark: `A` is set the sequence indicated in the table;
1 mark: `B` is set the sequence indicated in the table;
1 mark: `C` is set the sequence indicated in the table;
1 mark: `NoOfCats` is set to 5, `Cat[1]` is set to 1;
1 mark: `Cat[2]` is set to 2 and `Cat[3]` is set to 3;
1 mark: `Cat[4]` is set to 1 and `Cat[5]` is set to 1;

Info for examiner: Ignore the empty cells in the sequences - values do not need to be set in the rows indicated in the table.

6

(e) **Mark is for AO2 (analyse)**

To work out which cats will travel together to the show / /
To plan which cats will be in the van on which journey to the cat show / /
To colour the vertices of a graph / /
To create a decomposition of a graph;
Max 1

1

(f) **All marks AO1 (knowledge)**

1 mark (1 from): The problem can be solved / / algorithm exists for problem;
But it cannot be solved in polynomial time / / but not quickly enough to be useful;
Max 2
1 mark: It takes an unreasonable amount of time; to solve;
A Too long time but **R** Long time

2

(g) **All marks AO1 (understanding)**

1 mark: Use of heuristic; algorithm that makes a guess based on experience; That provides a close-to-optimal solution / approximation; that only works in some cases; **A** non-optimal

Example of heuristic method eg hill-climbing / stochastic / local improvement / greedy algorithms / simulated annealing / trial and error / any reasonable example;

1 mark: Relax some of the constraints on the solution; **A** Solve simpler version of problem

2

[16]

Q4.

(a)

Description	Correct letter (A-D)
A graph that is not connected.	B;
A graph that is a tree.	A;

If a letter has been used more than once then mark it as correct in the row that it is correct in, if any.

2

(b) **Example 1**

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	1	0	1	0	1	0
3	0	1	0	0	1	1
4	1	0	0	0	1	0
5	0	1	1	1	0	0
6	0	0	1	0	0	0

Example 2

	1	2	3	4	5	6
1	0	1	0	1	0	0
2		0	1	0	1	0
3			0	0	1	1
4				0	1	0
5					0	0
6						0

1 mark for labelling matrix with indices running from 1 to 6 on both axis and filled only with 0s and 1s, or some other symbol to indicate presence / absence of edge. e.g. T / F (allow a third symbol along diagonal). Absence can be represented by an empty cell.
1 mark for correct values entered into matrix, as shown in either example above

In **Example 2**, the shaded portion can be in either half – some indication must be made that half of the matrix is not being used. This could just be leaving it blank, unless the candidate has also represented absence of an edge by leaving cells blank.

Allow use of a third symbol in the central diagonal to indicate it unused, as it would not make sense to use it in this example.

Accept column and row labels in any order so long as they correspond to the data i.e. do not have to be in sequence 1 to 6.

2

(c)

						Discovered						Parent							
S	D	V	U	C	Queue	1	2	3	4	5	6	1	2	3	4	5	6	Found	Output
						F	F	F	F	F	F								
1	6				1	T	F	F	F	F	F							F	
1	6	1	2		2	T	T	F	F	F	F		1					F	
1	6	1	4		2 4	T	T	F	T	F	F		1		1			F	
1	6	2	1		4	T	T	F	T	F	F		1		1			F	
1	6	2	3		4 3	T	T	T	T	F	F		1	2	1			F	
1	6	2	5		4 3 5	T	T	T	T	T	F		1	2	1	2		F	
1	6	4	1		3 5	T	T	T	T	T	F		1	2	1	2		F	
1	6	4	5		3 5	T	T	T	T	T	F		1	2	1	2		F	
1	6	3	2		5	T	T	T	T	T	F		1	2	1	2		F	
1	6	3	5		5	T	T	T	T	T	F		1	2	1	2		F	
1	6	3	6		5 6	T	T	T	T	T	T		1	2	1	2	3	T	
1	6	3	6	6	5 6	T	T	T	T	T	T		1	2	1	2	3	T	6
1	6	3	6	3	5 6	T	T	T	T	T	T		1	2	1	2	3	T	3
1	6	3	6	2	5 6	T	T	T	T	T	T		1	2	1	2	3	T	2
1	6	3	6	1	5 6	T	T	T	T	T	T		1	2	1	2	3	T	1

1 mark for having the correct value changes in each region highlighted by a rectangle and no incorrect changes in the region. Ignore the contents of any cells that are not changed.
A Alternative indicators that clearly mean True and False.
A It is not necessary to repeat values that are already set (shown lighter in table)
A For the queue column, for cells that should only have one value in them, accept if the student has written out the value twice at both the Front and the Rear, so long as this has been done consistently throughout the table. eg "4" written as "4 4".

6

(d) So that packets / data arrive as quickly as possible / more quickly / with lower latency;
So that packets / data are delivered at lower cost / lowest cost;
So that routers do not have to process more packets / data due to unnecessary hops being made;
NE it is quicker without clarifying what "it" refers to
MAX 1

1

[11]

Q5.

(a) Omitting unnecessary details (from a representation) / / Storing only those details which are necessary (in the context of the problem being solved);
R responses that do not refer to abstraction in the context of data or modelling

1

(b) **SUBJECT MARKING POINTS:**

Representation as a graph:

Vertex / node represents a station; **A** junction between railway lines

Edge / arc / line represents a (direct) connection / railway line between two stations; **R** vector
Graph must be weighted / / edges have weights;
Distance between two stations must be written on edge / / stored with edge / / weights will be distances;
Could be more than one direct route between two stations; in which case shortest of the distances would be stored as the weight;

Graph would be undirected as trains can travel in both directions between each station;
OR
Graph would be directed as some lines may only be traversable in one direction;
Note: Only accept one of the above two points about whether the graph would be directed or undirected. Must have reason.

Implementation as array:

Each station assigned a (unique) number (to be used as array index); - **This mark available regardless of how the rest of the implementation is done**

Using an adjacency matrix:

The (adjacency) matrix would be a two-dimensional array (of numbers);
Array contains one row and one column for each station / / An n x n array is required to represent n stations; **A** rows and columns labelled with stations for BOD mark
If there is a (direct) connection between the two stations, store the distance between the two stations at the intersection of the row / column for the stations;
If there is no (direct) connection between the two stations, store a value to indicate this at the intersection of the row / column for the stations; **A** examples of values eg 0, ∞, NULL that could not be valid distances including any alphanumeric indicator

Using an adjacency list as a 2d array of numbers:

Adjacency list could be stored in a two-dimensional array (of records or similar);
In one dimension there would have to be n rows / columns for n stations / / one row / column per station;
In the other dimension the number of columns / rows would be determined by the highest degree of any vertex / / the maximum number of neighbours a vertex has / / the maximum number of (direct) connections that any station has;
If a station is (directly) connected to another station then in the row / column for the first station, a new entry (record) would be made consisting of the number of / / an identifier for the second station and the distance to it; **NE** just to state identifier, must have distance as well
Note: Also accept implementation in two two-dimensional arrays, with one storing the stations and the other the distances, as long as made clear station identifiers and distances stored in corresponding positions.

Using an adjacency list as a 1d array of strings:

Adjacency list could be stored in a (one-dimensional) array of strings;
One row per station;
The string for a station contains, for each station that it is connected to, the station identifier / name and distance;
Use of a delimiter between values;

REFER ANY OTHER WORKABLE SOLUTION TO A TEAM LEADER

If comparison made between adjacency matrix and adjacency list (not asked for):

Adjacency list might be more efficient (in terms of storage space) as graph is likely to be sparse / / as few edges between vertices / / as most stations only (directly) connected to a small number of other stations;
Adjacency matrix might be more efficient (in terms of speed) as shortest route finding algorithm is likely to need to lookup many distances when computing a route;

Note on use of diagrams: Candidates may choose to use diagrams to help clarify their responses. When marking, use may be made of such diagrams to help clarify understanding of the written description, however as this question assesses quality of written communication, marks should be awarded for the written description, not directly for the diagrams themselves.

8

HOW TO AWARD MARKS:

Mark Bands and Description

To achieve a mark in this band, candidates must meet the subject criterion (SUB) and all 5 of the quality of written communication criteria (QWCx).

SUB Candidate has covered the graph representation and array implementation in detail and all or almost all of the required detail for an implementation is present. The candidate has made at least seven subject-related points.
QWC1 Text is legible.
QWC2 There are few, if any, errors of spelling, punctuation and grammar. Meaning is clear.
QWC3 The candidate has selected and used a form and style of writing appropriate to the purpose and has expressed ideas clearly and fluently.
QWC4 Sentences (and paragraphs) follow on from one another clearly and coherently.
QWC5 Appropriate specialist vocabulary has been used.

7-8

To achieve a mark in this band, candidates must meet the subject criterion (SUB) and 4 of the 5 quality of written communication criteria (QWCx).

SUB Candidate has covered both the graph representation and array implementation, making some valid points for each, but the level of detail may not be sufficient to implement. The candidate has made at least five subject-related points.
QWC1 Text is legible.
QWC2 There may be occasional errors of spelling, punctuation and grammar. Meaning is clear.
QWC3 The candidate has, in the main, used a form and style of writing appropriate

to the purpose, with occasional lapses. The candidate has expressed ideas clearly and reasonably fluently.

QWC4 The candidate has used well-linked sentences (and paragraphs).

QWC5 Appropriate specialist vocabulary has been used.

5-6

To achieve a mark in this band, candidates must meet the subject criterion (SUB) and 4 of the 5 quality of written communication criteria (QWCx).

SUB Candidate has made some relevant points but the description is either lacking in detail or only covers one of the graph representation or array implementation.

QWC1 Most of the text is legible.

QWC2 There may be some errors of spelling, punctuation and grammar but it should still be possible to understand most of the response.

QWC3 The candidate has used a form and style of writing which has many deficiencies. Ideas are not always clearly expressed.

QWC4 Sentences (and paragraphs) may not always be well-connected.

QWC5 Specialist vocabulary has been used inappropriately or not at all.

1-4

Candidate has made no relevant points.

0

Note: Even if English is perfect, candidates can only get marks for the points made at the top of the mark scheme for this question.

If a candidate meets the subject criterion in a band but does not meet the quality of written communication criteria then drop mark by one band, providing that at least 4 of the quality of language criteria are met in the lower band. If 4 criteria are not met then drop by two bands.

[9]

Q6.

(a) **Connected** // There is a path between each pair of vertices;
Undirected // No direction is associated with each edge;
Has no cycles // No (simple) circuits // No closed chains // No closed paths in which all the edges are different and all the intermediate vertices are different // No route from a vertex back to itself that doesn't use an edge more than once or visit an intermediate vertex more than once;
A no loops
Alternative definitions:
A simple cycle is formed if any edge is added to graph;
Any two vertices can be connected by a unique simple path;

Max 1

(b) No route from entrance to exit / through maze;
Maze contains a loop/circuit ;
A more than one route through maze;
Part of the maze is inaccessible / enclosed;
R Responses that clearly relate to a graph rather than the maze

Max 1

(c)

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	1	0	1	1	0	0	0
3	0	1	0	0	0	0	0
4	0	1	0	0	1	0	0
5	0	0	0	1	0	1	1
6	0	0	0	0	1	0	0
7	0	0	0	0	1	0	0

(allow some symbol in the central diagonal to indicate unused)

or

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2		0	1	1	0	0	0
3			0	0	0	0	0
4				0	1	0	0
5					0	1	1
6						0	0
7							0

(with the shaded portion in either half – some indication must be made that half of the matrix is not being used. This could just be leaving it blank, unless the candidate has also represented absence of an edge by leaving cells blank)

1 mark for drawing a 7x7 matrix, labelled with indices on both axis and filled only with 0s and 1s, or some other symbol to indicate presence/absence of edge. e.g. T/F. Absence can be represented by an empty cell.
1 mark for correct values entered into matrix, as shown above;

2

(d) (i) Routine defined in terms of itself // Routine that calls itself;
A alternative names for routine e.g. procedure, algorithm
NE repeats itself

1

(ii) Stores return addresses;
Stores parameters;
Stores local variables; NE temporary variables
Stores contents of registers;
A To keep track of calls to subroutines/methods etc.

Max 1

Procedures / invocations / calls must be returned to in reverse order (of being called);
As it is a LIFO structure;
A FILO
As more than one / many return addresses / sets of values may need to be stored (at same time) // As the routine calls itself and for each call/invocation a new return address / new values must be stored;

Max 1

2

(e)

				Discovered							Completely Explored							
Call	V	U	EndV	1	2	3	4	5	6	7	1	2	3	4	5	6	7	F
	-	-	7	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
DFS(1,7)	1	2	7	**T**	F	F	F	F	F	F	F	F	F	F	F	F	F	F
DFS(2,7)	2	1	7	T	**T**	F	F	F	F	F	F	F	F	F	F	F	F	F
		3	7	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F
DFS(3,7)	3	2	7	T	T	**T**	F	F	F	F	F	F	**T**	F	F	F	F	F
DFS(2,7)	2	4	7	T	T	T	F	F	F	F	F	F	T	F	F	F	F	F
DFS(4,7)	4	2	7	T	T	T	**T**	F	F	F	F	F	T	F	F	F	F	F
		5	7	T	T	T	T	F	F	F	F	F	T	F	F	F	F	F
DFS(5,7)	5	4	7	T	T	T	T	**T**	F	F	F	F	T	F	F	F	F	F
		6	7	T	T	T	T	T	F	F	F	F	T	F	F	F	F	F
DFS(6,7)	6	5	7	T	T	T	T	T	**T**	F	F	F	T	F	F	**T**	F	F
DFS(5,7)	5	7	7	T	T	T	T	T	T	F	F	F	T	F	F	T	F	F
DFS(7,7)	7	5	7	T	T	T	T	T	T	**T**	F	F	T	F	F	T	**T**	**T**
DFS(5,7)	5	-	7	T	T	T	T	T	T	T	F	F	T	F	**T**	T	T	T
DFS(4,7)	4	-	7	T	T	T	T	T	T	T	F	F	T	**T**	T	T	T	T
DFS(2,7)	2	-	7	T	T	T	T	T	T	T	F	**T**	T	T	T	T	T	T
DFS(1,7)	1	-	7	T	T	T	T	T	T	T	**T**	T	T	T	T	T	T	T

1 mark for having the correct values changes in each region highlighted by a rectangle and no incorrect changes in the region. Ignore the contents of any cells that are not changed.

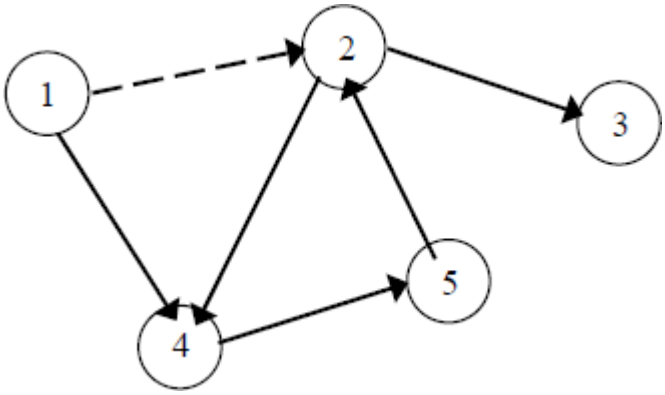
A alternative indicators that clearly mean True and False.
A it is not necessary to repeat values that are already set (shown lighter in table)

5

[12]

Q7.

(a)



1 mark for all 5 lines correctly drawn
1 mark for all 5 arrowheads pointing in correct directions
Max 1 if more than 5 lines drawn by candidate (note that dotted arrow is given in question)

A arrowheads at any position on line

2

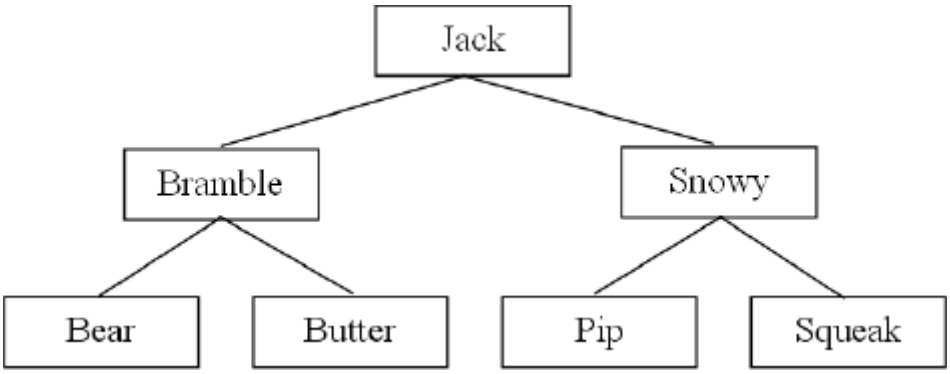
(b) Adjacency matrix appropriate when there are many edges between vertices // when edges may be frequently changed // when presence/absence of specific edges needs to be tested (frequently)
Adjacency list appropriate when there are few edges between vertices // when graph is sparse // when edges rarely changed //when presence/absence of specific edges does not need to be tested (frequently)
A alternative words which describe edge e.g. connection, line

2

(c) Connected // There is a path between each pair of vertices;
Undirected // No direction is associated with each edge;
Has no cycles // No (simple) circuits // No closed chains // No closed paths in which all the edges are different and all the intermediate vertices are different // No route from a vertex back to itself that doesn't use an edge more than once or visit an intermediate vertex more than once;
Alternative definitions:
Graph with no cycles, and a simple cycle is formed if any edge is added to it;;
Graph which is connected, and it is not connected anymore if any edge is removed from it;;
Graph in which any two vertices can be connected by a unique simple path;;
(Note: not just adjacent vertices)
Graph which is connected and has n - 1 edges where n is no of vertices;;
Graph which has no simple cycles and has n - 1 edges where n is no of vertices;;

Max 2

(d)



1 mark for Jack as root
1 mark for Bramble and Snowy as children of Jack
1 mark for four correct children of Bramble and Snowy

DPT if arrows drawn instead of lines
DPT if any node has more than 2 child nodes
A "mirror image" answers which are consistent.

3

(e) **For solution with 3 arrays:**
One array stores data items;
One array for left child pointers;
One array for right child pointers;
Pointers stored at same location in arrays as corresponding data item;
For solution with 1 array of records:
Record created to store data item and pointers;
One pointer to left child;
One pointer to right child;
For either of the above solutions:
Rogue value (allow example) used to indicate no child;
Variable indicates position in array(s) of root node // Root node stored at first location/start of array(s);
If answered as diagram:
Column for data with at least three correct data items in it;
Use of rogue value for a node that does not have child;
Correct value for a start pointer variable indicating position of root node in the array (not drawn as an arrow, array indices must be labelled);
Column for left child pointers*;
Column for right child pointers*;
* = To get these marks, there must be a sufficient number of pointers to demonstrate that the data structure is a representation of a binary tree, but it is not necessary for every item to be shown. Also the array indices must be shown.

Max 3

[12]

Q1.

In previous years there have been questions asking students to complete an adjacency matrix based on a diagram of a graph and most students were able to answer question (a) this year. This was the first time that an adjacency matrix for a weighted graph had been asked for and some students had clearly not seen this type of question before and only included an indicator that there was an edge between two nodes rather than the weight of the edge between the two nodes; this meant they only got one of the two marks available for this question.

Questions (b)-(d) were about graph theory. Question (c) was well-answered with students identifying that it was not a tree because there were cycles. The most common incorrect answer was to say that it wasn't a tree because the edges have weights associated with them. Question (d) was also well-answered. Answers to (b) often showed that students were not as familiar with adjacency lists as they are with adjacency matrices.

For question (e) students had to complete a trace of Djikstra's Algorithm. This topic was not on the previous A-level specification and was often poorly answered suggesting many students had not tried to complete a trace for the algorithm before. For question (f) many students gave an answer that explained the point of Djikstra's Algorithm (find the shortest route from a node to each other node) rather than what the specific output in the algorithm given in the question would be (the distance of the shortest route from node 1 to node 6).

Q2.

This question was about data structures, with much of the emphasis placed on binary trees.

Parts (a) and (b), which related to searching a binary search tree, were both well answered with two thirds of candidates correctly identifying the items that would be examined during the search and over half correctly identifying the time complexity of the search operation for part (b).

For part (c) candidates had to represent a binary tree using an array of records. This was well tackled with candidates correctly using pointers to indicate the relationships between the data items. It was not enough to represent leaf node branches with a blank space or a dash for the left and right pointers. An appropriate value such as an unused index number eg 0 or a NULL value was required.

Most candidates achieved a reasonable number of marks for question part (d). The key difference between a static and dynamic data structure, that the former had a fixed size that was defined at compile time and that the latter had a variable size which could change at run time was well understood. However, not many candidates went on to explain any other differences, such as the fact that memory space might be wasted if a static structure was relatively empty or that a static structure would generally be allocated consecutive memory locations. The purpose of the heap was well understood, as being a pool of available unused memory that could be allocated to a dynamic structure at runtime. The most common misunderstanding was that the heap was where a dynamic structure stored its data. Some candidates made did not get the mark for explaining the purpose of the heap as whilst they made clear that the heap was used for dynamically allocating memory, their responses did not make clear that the heap was the unused memory rather than the memory that new data was stored in.

Part (e) (i) was well answered with the majority of candidates correctly identifying the

order that the items would be output. Virtually everyone who correctly identified the order explained the significance of this for part (e) (ii).

For part (f) candidates had to explain why graph traversal was a more complex problem than tree traversal. Many responses recognised that features such as cycles and weighting were the key factors that might contribute to this. Some candidates went beyond what was required and provided excellent explanations of, for example, how a cycle in a graph might cause a problem for a traversal algorithm.

Q4.

This question was about graphs and graph traversal in the context of a computer network.

Part (a) required students to identify graphs with specific properties. Almost all students were able to achieve one of the two marks and just over a third achieved both. Part (b) was also well tackled, with over 90% of students achieving both marks for correctly completing the adjacency matrix.

For part (c) students had to trace an algorithm for performing a breadth first search of a graph. This was well tackled, with over half of students achieving at least four of the six marks. The most commonly made mistakes were to incorrectly record the queue contents and to update the Parent array incorrectly.

For part (d) students needed to explain why finding the shortest route was a useful thing to do in the context of the question. Just over half of students correctly explained that this would allow data to be transmitted more quickly or would reduce congestion. Some students missed out on achieving a mark by giving responses that were not in context.

Q5.

(a) More than half of the candidates were able to correctly define abstraction in this part. The question was asked in the context of data representation, so an appropriate definition would have been to store only those details of a problem / model that were required in the context of the problem being solved. Common mistakes were to state that unnecessary complexity would be hidden from the user, rather than being removed from the model altogether, or to define a simulation instead of abstraction.

(b) This part was the question that assessed quality of written communication. As such, it was particularly important the candidates used technical terms accurately when writing their responses. Almost all candidates covered both aspects of the question: the representation as a graph and the representation using arrays as an adjacency matrix or list.

In almost all responses, the graph part of the answer was better than the array representation part, with most students being able to correctly explain how the underground railway network could be represented as a graph. Some students lost marks by not using the correct terminology or by drawing diagrams but not explaining points such as that a station would be represented as a node.

As the question paper stated, diagrams needed to be fully explained if they were used. A common mistake was to state that stations would be represented by nodes and that railway tracks would be represented by vertices; vertex in a synonym for node but was being mistakenly used instead of the term edge or arc. A small but not insignificant number of students misinterpreted the term graph and explained how the network might be represented as a bar chart or scatter graph.

The part of the question relating to the representation using arrays as an adjacency matrix or list was poorly tackled. Many students discussed how the representation could be drawn out as a grid or list on paper instead of how it could be represented using arrays in a programming language, or described solutions that were not really either an adjacency matrix or list.

Those who decided to represent the data as an adjacency matrix often suggested that the station names would be written into the array rather than that stations would be associated with a number that would be used as an index into the array. That said, pleasingly, most students who suggested a matrix representation recognised that the distances would be filled into the array at the appropriate location and that a null value would be used where there was no route. Some students used an array containing only 0s and 1s instead of distances.

Students who chose to present an adjacency list solution rarely explained how this would be achieved in a programming language, focussing instead on how it could be depicted diagrammatically. Another common mistake was to talk about pointers, without explaining how these would work in the context of the array.

Q6.

Part (a): Two thirds of students were able to identify one property that a graph must have to be a tree. A small number confused a tree with a rooted tree and made assertions such as that a tree must have a root, which is incorrect.

Part (b): This question part tested students' understanding of the method being used to represent a maze as a graph. The majority of students correctly identified a feature of the maze that would stop its graph being a tree. The most commonly seen correct response identified that there could be a loop in the maze. Other possibilities included that part of the maze could be inaccessible or that part of the maze might only be traversable in one direction. Some students failed to achieve the mark because they re-answered part (a), discussing a feature of a graph that would stop it being a tree, rather than a feature of a maze.

Part (c): Students were asked to represent the graph of the maze as an adjacency matrix. Three quarters of students scored both marks for this question part. Responses where symbols other than 0s and 1s were used in the matrix were accepted, as long as they could be viewed as an accurate representation of the graph.

Part (d)(i): The vast majority of students were able to identify that a recursive routine would call itself. A small number asserted that a recursive routine would repeat itself, which was not considered to be enough for a mark as this could equally have been a description of iteration.

Part (d)(ii): Most students scored some marks for this question part, but less than a fifth achieved both. The most widely understood point was that the data would need to be removed from the stack in the reverse of the order that it was put onto it so that the recursion could be unwound. Less well understood was the types of data that would be stored, such as return addresses and local variables.

Part (e): Most students achieved some marks on this question part and around a quarter achieved all five for a fully complete trace. The most commonly made mistake was to update, incorrectly, the Completely Explored array as the recursive calls were made, as opposed to when the recursion unwound.

Q7.

Part (a): The use of the adjacency matrix was clearly well understood with all but a few candidates achieving full marks.

Part (b): There were some good responses to this question part, but also quite a lot of confused answers. An adjacency matrix is more appropriate when there are many edges in a graph, or if these edges need to be checked or updated frequently. An adjacency list is appropriate for graphs with few edges (sparse graphs) or where the edges are not checked / updated frequently.
Neither the number of vertices in a graph nor the available memory would influence the choice.
There was confusion over the use of terminology with some candidates apparently using the term vertex to mean edge.

Part (c): This question part was poorly answered, with only a third of candidates scoring any marks. A tree is a graph that is connected, undirected and has no cycles. Some candidates gave responses that referred only to specific types of tree, either rooted trees or binary trees.
These responses did not gain credit.

Part (d): The vast majority of candidates knew how to construct a binary search tree. The most common cause of error appeared to be candidates forgetting the order of the letters in the alphabet rather than forgetting the principles that should be used to construct the tree. A small number of candidates mistakenly drew arrows instead of lines between nodes.

Part (e): There were some good responses to this question part but many were disappointing and a surprising number of candidates did not write a response at all. Many candidates who did answer chose to use a diagram to illustrate their response which was quite acceptable, so long as the diagram included enough detail to make clear that it was a representation of a binary tree.