2.4 Graphs

Name: ____________________

Class: ____________________

Date: ____________________

Time: **92 minutes**

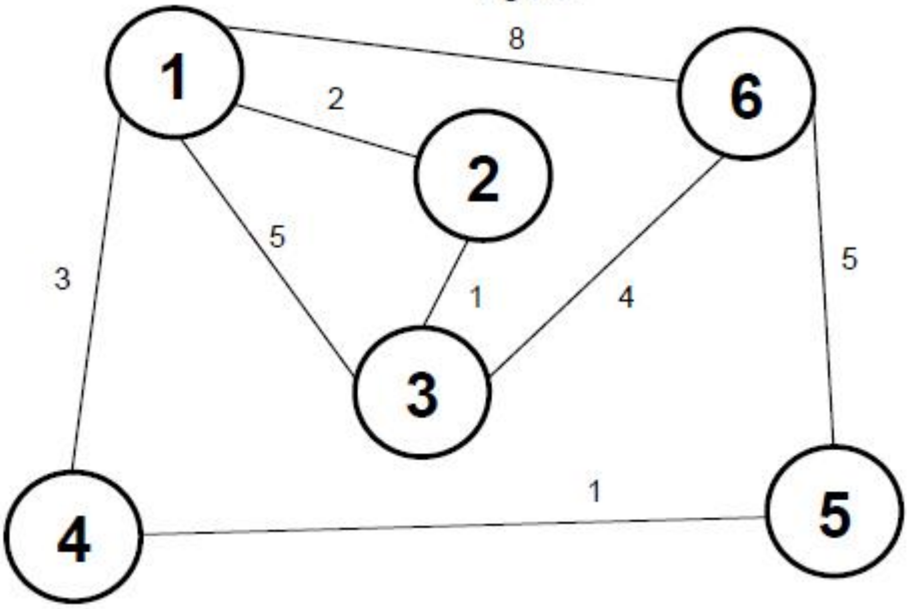
Marks: **67 marks**

Comments:

Q1.

Figure 1 is a graph that shows the time it takes to travel between six locations in a warehouse. The six locations have been labelled with the numbers 1 - 6. When there is no edge between two nodes in the graph this means that it is not possible to travel directly between those two locations. When there is an edge between two nodes in the graph the edge is labelled with the time (in minutes) it takes to travel between the two locations represented by the nodes.

Figure 1



(a) The graph is represented using an adjacency matrix, with the value 0 being used to indicate that there is no edge between two nodes in the graph.

A value should be written in every cell.

Complete the unshaded cells in **Table 1** so that it shows the adjacency matrix for **Figure 1**.

Table 1

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

(2)

(b) Instead of using an adjacency matrix, an adjacency list could be used to represent the graph. Explain the circumstances in which it would be more appropriate to use an adjacency list instead of an adjacency matrix.

____

____

____

____

(2)

(c) State **one** reason why the graph shown in **Figure 1** is **not** a tree.

____

____

(1)

(d) The graph in **Figure 1** is a weighted graph. Explain what is meant by a **weighted graph**.

____

____

(1)

Figure 2 contains pseudo-code for a version of Djikstra's algorithm used with the graph in **Figure 1**.

`Q` is a priority queue which stores nodes from the graph, maintained in an order based on the values in array `D`. The reordering of `Q` is performed automatically when a value in `D` is changed.

`AM` is the name given to the adjacency matrix for the graph represented in **Figure 1**.

Figure 2

```
Q ← empty queue
FOR C1 ← 1 TO 6
  D[C1] ← 20
  P[C1] ← -1
  ADD C1 TO Q
ENDFOR
D[1] ← 0
WHILE Q NOT EMPTY
  U ← get next node from Q
  remove U from Q
  FOR EACH V IN Q WHERE AM[U, V] > 0
    A ← D[U] + AM[U, V]
    IF A < D[V] THEN
      D[V] ← A
      P[V] ← U
```

```
      ENDIF
    ENDFOR
  ENDWHILE
  OUTPUT D[6]
```

(e) Complete the unshaded cells of **Table 2** to show the result of tracing the algorithm shown in **Figure 2**. Some of the trace, including the maintenance of Q, has already been completed for you.

Table 2

				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													
		3													
		4													
		6													
2	3,4,5,6	3													
3	4,5,6	6													
4	5,6	5													
5	6	6													
6	-														

(7)

(f) What does the output from the algorithm in **Figure 2** represent?

____

____

(1)

(g) The contents of the array P were changed by the algorithm. What is the purpose of the array P?

____

____

____

____

(2)

(Total 16 marks)

Q2.

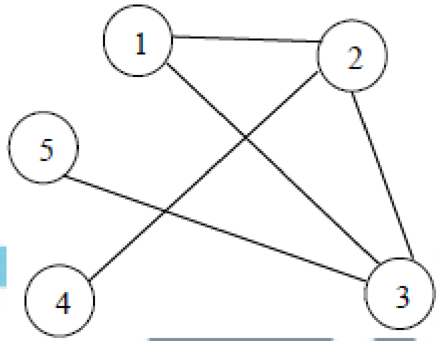
The Cat transportation company (CTC) is a business that specialises in preparing cats for cat shows.

They need to take five cats to the AQA cat show. They will transport the cats in their van. CTC owns only one van.

They cannot put all the cats in their van at the same time because some of the cats get stressed when in the company of some of the other cats. The cats would not therefore arrive in top condition for the cat show if they were all in the van at the same time.

The graph in **Figure 1** shows the relationships between the five cats (labelled 1 to 5). If there is an edge between two cats in the graph then they **cannot** travel in the van together at the same time.

Figure 1



(a) Explain why the graph in **Figure 1** is **not** a tree.

____

____

(1)

(b) Represent the graph shown in **Figure 1** as an adjacency list by completing **Table 1**

Table 1

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

(2)

(c) **Table 2** shows how the graph in **Figure 1** can be represented as an adjacency matrix.

Table 2

Vertex (in Figure 1)	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	1	0
3	1	1	0	0	1
4	0	1	0	0	0
5	0	0	1	0	0

Explain the circumstances in which it is more appropriate to represent a graph using an adjacency list instead of an adjacency matrix.

____

____

____

____

(2)

(d) **Figure 2** shows an algorithm, written in pseudo-code, that CTC use.

Figure 2

```
NoOfCats ← 5
Cat[1] ← 1
FOR A ← 2 TO NoOfCats
  B ← 1
  C ← 1
  WHILE B < A DO
    IF M[A, B] = 1
      THEN
        IF Cat[B] = C
          THEN
            B ← 1
            C ← C + 1
          ELSE B ← B + 1
        ENDIF
      ELSE B ← B + 1
    ENDIF
  ENDWHILE
  Cat[A] ← C
ENDFOR
```

The two-dimensional array, `M`, is used to store the adjacency matrix shown in **Table 2**.

Complete **Table 3** to show the result of tracing the algorithm in **Figure 2**.

Table 3

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

(6)

(e) Explain the purpose of the algorithm in **Figure 2**.

___

___

(1)

(f) After a cat show, CTC needs to return the cats to their owners. They can have all the cats in the van at the same time because the show is now finished.

CTC likes to plan the return journey so that the shortest possible distance is travelled by the van. This is an example of an intractable problem.

What is meant by an intractable problem?

___

___

___

___

(2)

(g) What approach might a programmer take if asked to solve an intractable problem?

___

___

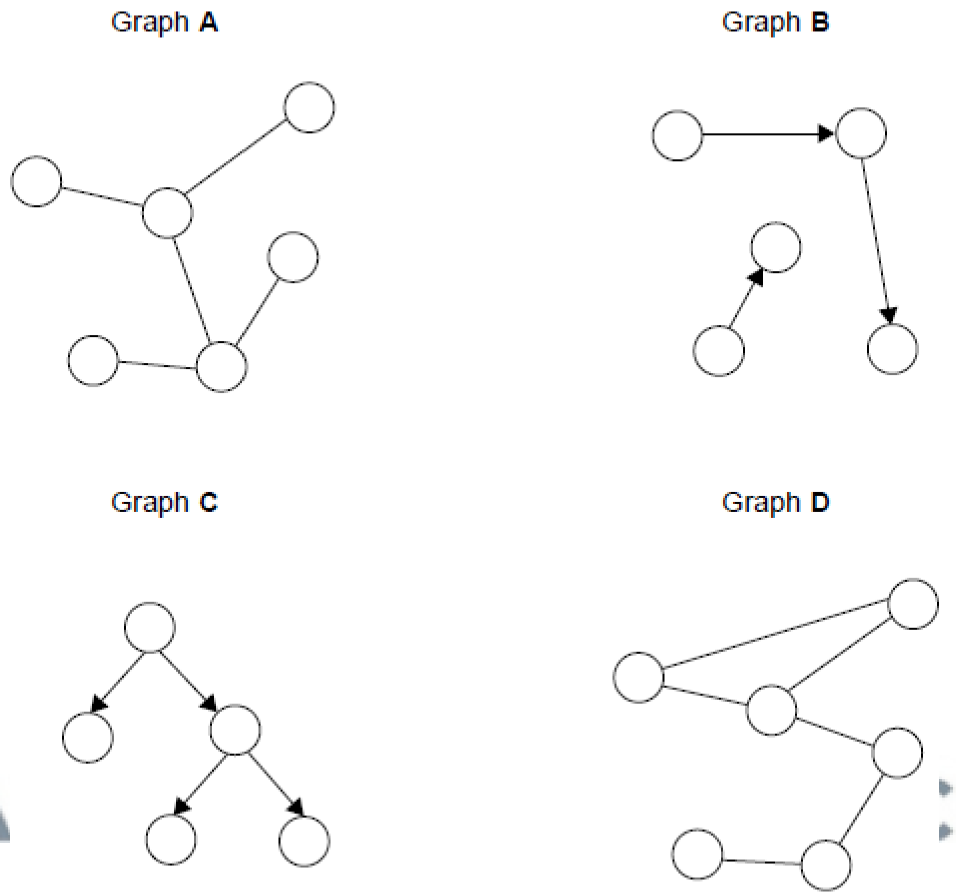
(2)

(Total 16 marks)

Q3.

(a) **Figure 1** shows four graphs, labelled with the letters **A** to **D**.

Figure 1



Complete **Table 1** below. In the **Correct letter (A-D)** column write the appropriate letter from **A** to **D** to indicate which graph in **Figure 1** matches the description in the **Description** column.

Do **not** use the same letter more than once. You will not need to use all of the letters.

Table 1

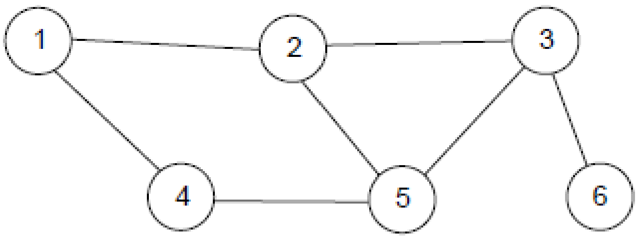
Description	**Correct letter (A-D)**
A graph that is not connected	
A graph that is a tree	

(2)

(b) It is possible to represent a computer network as a graph, with each vertex representing a router and each edge representing a communications link.

Figure 2 is a graph representation of a medium-sized computer network that consists of 6 routers and 7 communications links. The routers have been numbered from 1 to 6.

Figure 2



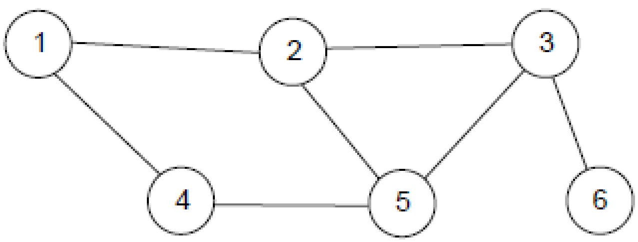
Complete **Table 2** below to show how the graph in **Figure 2** would be stored using an adjacency matrix.

Table 2

(2)

(c) **Figure 2** is repeated here.

Figure 2 (Repeated)



A simple method of determining the shortest path through a network from one router to another is to perform a breadth first search of the graph representation of the network.

The algorithm in **Figure 3** can be used to perform a breadth first search of a graph. It makes use of two subroutines, `PutVertexIntoQueue` and `GetVertexFromQueue`, which are explained below the algorithm.

Figure 3

```
Procedure ShortestRoute(S, D)
    PutVertexIntoQueue(S)
    Discovered[S] ← True
    Found ← False
    While Queue is Not Empty And Found = False Do
          V = GetVertexFromQueue
          For each vertex U which is adjacent to V Do
              If Discovered[U] = False And Found = False Then
                PutVertexIntoQueue(U)
                Discovered[U] ← True
                Parent[U] ← V
                If U = D Then Found ← True
              EndIf
          EndFor
    EndWhile
    If Found = True Then
          C ← D
          Output D
          Repeat
              C ← Parent[C]
              Output C
          Until C = S
    EndIf
EndProcedure
```

- `PutVertexIntoQueue` is a subroutine that adds a vertex to the rear of a queue.
- `GetVertexFromQueue` is a subroutine that returns the name of the vertex at the front of the queue and removes it from the queue.

Complete the trace table below to show how the `Discovered` and `Parent` arrays, the variable `Found` and the queue contents are updated, together with what output is produced by the algorithm when it is called using `ShortestRoute(1, 6)`.

Before the algorithm is carried out, all cells in the `Discovered` array are set to the value `False` and the queue is empty.

The values of the variables `S, D, V, U` and `C` have already been entered into the table for you.

The letter `F` has been used as an abbreviation for `False`. You should use T as an abbreviation for `True`.

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

(6)

(d) Explain why it is useful to find the shortest path through the network.

____

____

(1)
(Total 11 marks)

Q4.

A graph can be drawn to represent a maze. In such a graph, each graph vertex represents one of the following:

- the entrance to or exit from the maze
- a place where more than one path can be taken
- a dead end.

Edges connect the vertices according to the paths in the maze.

Diagram 1 shows a maze and **Diagram 2** shows one possible representation of this maze.

Position 1 in **Diagram 1** corresponds to vertex 1 in **Diagram 2** and is the entrance to the maze. Position 7 in **Diagram 1** is the exit to the maze and corresponds to vertex 7.

Dead ends have been represented by the symbol in **Diagram 2**.

Diagram 3 shows a simplified undirected graph of this maze with dead ends omitted.

Diagram 1

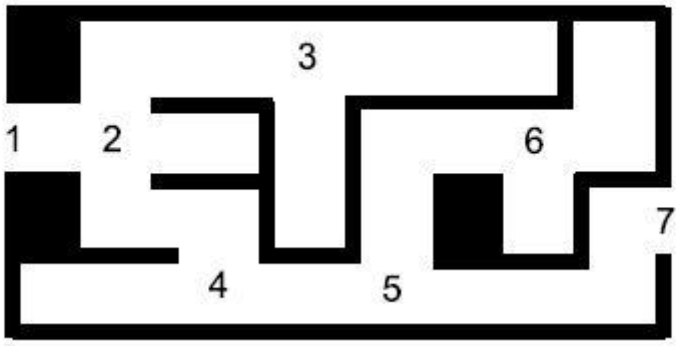
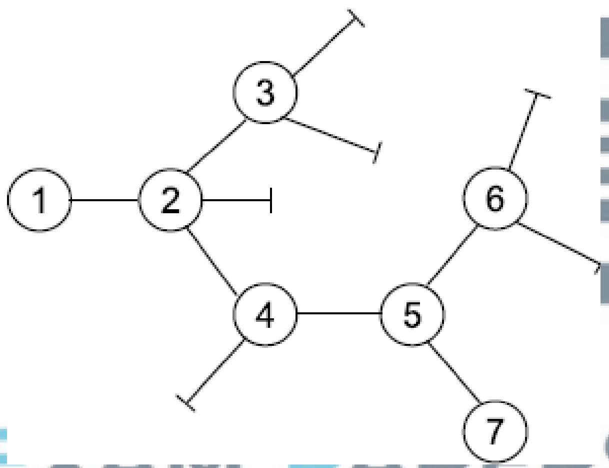
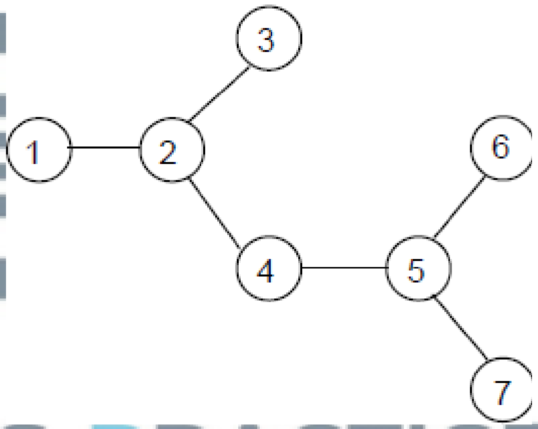


Diagram 2



Representation of maze including dead ends

Diagram 3



Graph representing maze with dead ends omitted

(a) The graph in **Diagram 3** is a tree.

State **one** property of the graph in **Diagram 3** that makes it a tree.

____

____

(1)

(b) The graphs of some mazes are not trees.

Describe a feature of a maze that would result in its graph **not** being a tree.

____

____

(1)

(c) Complete the table below to show how the graph in **Diagram 3** would be stored using an adjacency matrix.

(2)

(d) (i) What is a *recursive routine?*

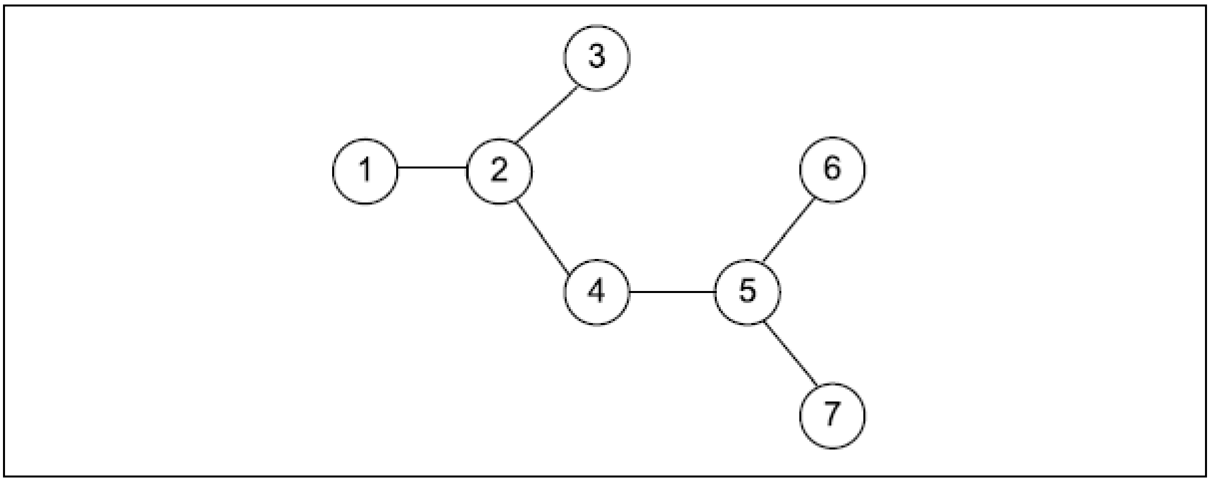
(1)

(ii) To enable the use of recursion a programming language must provide a stack.

Explain what this stack will be used for and why a stack is appropriate.

(2)

Diagram 3 is repeated here so that you can answer Question (e) without having to turn pages.



(e) A recursive routine can be used to perform a depth-first search of the graph that represents the maze to test if there is a route from the entrance (vertex 1) to the exit (vertex 7).

The recursive routine in the diagram below is to be used to explore the graph in **Diagram 3**. It has two parameters, `V` (the current vertex) and `EndV` (the exit vertex).

```
Procedure DFS(V, EndV)
    Discovered[V] ← True
    If V = EndV Then Found ← True
    For each vertex U which is connected to V Do
        If Discovered [U] = False Then DFS(U, EndV)
    EndFor
    CompletelyExplored[V] ← True
EndProcedure
```

Complete the trace table below to show how the `Discovered` and `CompletelyExplored` flag arrays and the variable `Found` are updated by the algorithm when it is called using `DFS(1,7)`.

The details of each call and the values of the variables `V`, `U` and `EndV` have already been entered into the table for you. The letter `F` has been used as an abbreviation for `False`. You should use `T` as an abbreviation for `True`.

				Discovered							CompletelyExplored							
Call	V	U	EndV	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Found
	-	-		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
DFS(1,7)	1	2	7															
DFS(2,7)	2	1	7															
		3	7															
DFS(3,7)	3	2	7															
DFS(2,7)	2	4	7															
DFS(4,7)	4	2	7															
		5	7															
DFS(5,7)	5	4	7															
		6	7															
DFS(6,7)	6	5	7															
DFS(5,7)	5	7	7															
DFS(7,7)	7	5	7															
DFS(5,7)	5	-	7															
DFS(4,7)	4	-	7															
DFS(2,7)	2	-	7															
DFS(1,7)	1	-	7															

(5)

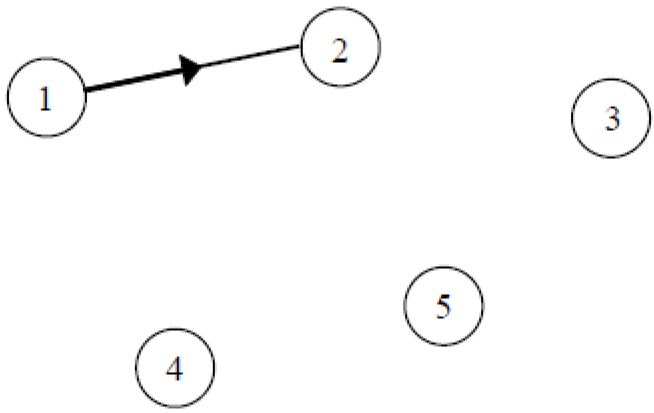
(Total 12 marks)

Q5.

The table below shows an adjacency matrix representation of a directed graph (digraph).

		1	**2**	**3**	**4**	**5**
From	**1**	0	1	0	1	0
	2	0	0	1	1	0
	3	0	0	0	0	0
	4	0	0	0	0	1
	5	0	1	0	0	0

(a) Complete this unfinished diagram of the directed graph.



(2)

(b) Directed graphs can also be represented by an adjacency list.

Explain under what circumstances an adjacency matrix is the most appropriate method to use to represent a directed graph, and under what circumstances an adjacency list is more appropriate.

____

____

____

(2)

(c) A tree is a particular type of graph.

What properties must a graph have for it to be a tree?

____

____

____

(2)

(d) Data may be stored as a binary tree.

Show how the following data may be stored as a binary tree for subsequent processing in alphabetic order by drawing the tree. Assume that the first item is the root of the tree and the rest of the data items are inserted into the tree in the order given.

Data items: Jack, Bramble, Snowy, Butter, Squeak, Bear, Pip

(3)

(e) A binary tree such as the one created in part (d) could be represented using one array of records or, alternatively, using three one-dimensional arrays.

Describe how the data stored in the array(s) could be structured for **one** of these two possible methods of representation.

____

____

____

____

____

____

____

____

(3)

(Total 12 marks)