Section A

Write your responses to the questions in Section A **on the question paper**.

You are advised to spend no longer than 40 minutes on this section.

---

1 The binary search method can be used to search for an item in an ordered list.

(a) Show how the binary search method works by writing numbers in the table below to indicate which values would be examined to determine if the name “Richard” appears in the list.

Write the number “1” by the first value to be examined, “2” by the second value to be examined and so on.

Position	Value	Order Examined In
1	Adam	
2	Alex	
3	Anna	
4	Hon	
5	Mohammed	
6	Moonis	
7	Niraj	
8	Philip	
9	Punit	
10	Ravi	
11	Richard	
12	Timothy	
13	Tushara	
14	Uzair	
15	Zara	

(3)

(b) Tick **one** box to indicate the order of time complexity of the binary search method.

Order of time complexity	**Tick one box**
$O(\log_2 n)$	
$O(n)$	
$O(n^2)$	

(1)

(c) A different list contains 137 names.

What is the maximum number of names that would need to be accessed to determine if the name "Rachel" appears in the list?

Write your answer in the box below.

(1)

(Total 5 marks)

2 A computer program is being developed to play a card game on a smartphone. The game uses a standard deck of 52 playing cards. The cards are placed in a pile on top of each other.

The cards will be dealt (ie given out) to players from the top of the deck.

When a player gives up a card it is returned to the bottom of the deck.

(a) Explain why a queue is a suitable data structure to represent the deck of cards in this game.

____

____

(1)

(b) The queue representing the deck of cards will be implemented as a **circular queue** in a fixed-size array named `DeckQueue`. The array `DeckQueue` has indices running from `1` to `52`.

Figure 1 shows the contents of the `DeckQueue` array and its associated pointers at the start of a game. The variable `QueueSize` indicates how many cards are currently represented in the queue.

Figure 1

`DeckQueue`

Index	Data
[1]	10-Spades
[2]	2-Hearts
[3]	King-Clubs
[4]	Ace-Hearts
.	
.	
.	
[52]	8-Clubs

`FrontPointer = 1`

`RearPointer = 52`

`QueueSize = 52`

(i) Twelve cards are dealt from the top of the deck.

What values are now stored in the `FrontPointer` and `RearPointer` pointers and the `QueueSize` variable?

`FrontPointer` = __________ `RearPointer` = __________

`QueueSize` = __________

(1)

(ii) Next, a player gives up three cards and these are returned to the deck.

What values are now stored in the `FrontPointer` and `RearPointer` pointers and the `QueueSize` variable?

`FrontPointer` = __________ `RearPointer` = __________

`QueueSize` = __________

(1)

(c) Write a pseudo-code algorithm to deal a card from the deck.

Your algorithm should output the value of the card that is to be dealt and make any required modifications to the pointers and to the `QueueSize` variable.

It should also cope appropriately with any situation that might arise in the `DeckQueue` array whilst a game is being played.

(6)
(Total 9 marks)

3 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.

Figure 2 shows a maze and **Figure 3** shows one possible representation of this maze.

Position 1 in **Figure 2** corresponds to vertex 1 in **Figure 3** and is the entrance to the maze. Position 7 in **Figure 2** is the exit to the maze and corresponds to vertex 7. Dead ends have been represented by the symbol ——┤ in **Figure 3**.

Figure 4 shows a simplified undirected graph of this maze with dead ends omitted.

Figure 2

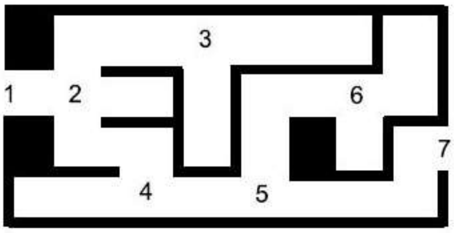
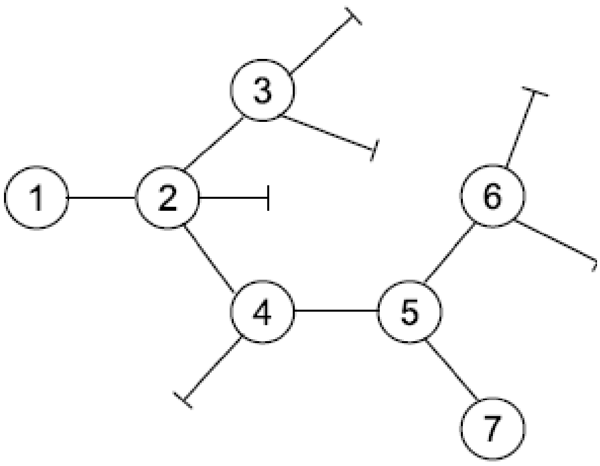
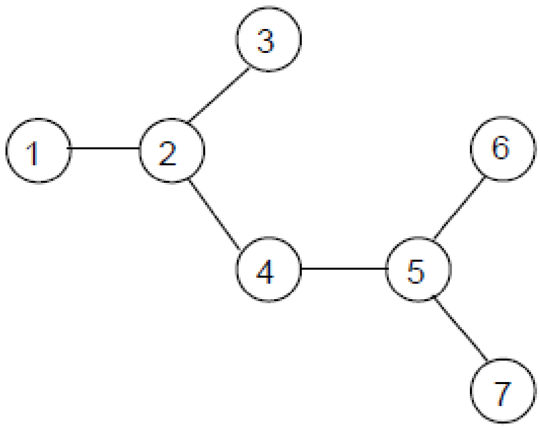


Figure 3



Representation of maze including dead ends

Figure 4



Graph representing maze with dead ends omitted

(a) The graph in **Figure 4** is a tree.

State **two** properties of the graph in **Figure 4** that makes it a tree.

____

____

____

____

(2)

(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 **Figure 4** would be stored using an adjacency matrix.

(2)

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

____

(1)

(ii) Programming languages use a stack to enable the use of recursion.

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

____

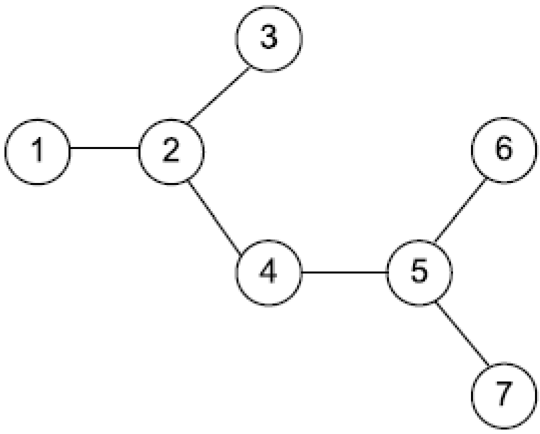
____

____

____

(2)

Figure 4 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 **Figure 4**. 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 13 marks)

4 Explain **two** differences between a dynamic data structure and a static data structure.

Difference 1: ____

____

Difference 2: ____

____

(2)
(Total 2 marks)

5 The contents of an array `Scores` are shown in the table below.

A pseudo-code representation of an algorithm is given below the table.

Scores							
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
18	23	36	21	58	40	45	59

```
Max ← 8
FOR Count1 ← 1 TO (Max – 1) DO
  FOR Count2 ← 1 TO (Max – 1) DO
    IF Scores[Count2] > Scores[Count2 + 1]
      THEN
      Temp ← Scores[Count2]
      Scores[Count2] ← Scores[Count2 + 1]
      Scores[Count2 + 1] ← Temp
    ENDIF
  ENDFOR
ENDFOR
```

(a) **One pass** is made through the outer loop of the algorithm in the diagram above.

Complete **Table 1** to show the changed contents of the array `Scores` after this single pass. You may use **Table 2** to help you work out your answer, though you are not required to use **Table 2**.

Table 1

Scores							
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

Table 2

Max	Count1	Count2	Temp	Scores							
				[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
				18	23	36	21	58	40	45	59

(4)

(b) What is the name of the standard algorithm shown in the pseudo-code above?

____

(1)

(c) Tick **one** box in **Table 3** to indicate the correct Order of **Time** Complexity of the standard algorithm you have just traced.

Table 3

Order of Time Complexity	Tick one box
O(n)	
$O(n^2)$	
$O(2^n)$	

(1)
(Total 6 marks)

6 (a) Complete the missing parts of the question posed by the Halting problem in the diagram below.

> Is it possible in general to ____ that
>
> can tell, given any program and its inputs and without
>
> ______________________________________, whether the given program with
>
> its given inputs will halt?

(2)

(b) What is the significance of the Halting problem?

____

____

(1)
(Total 3 marks)

Section B

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

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

---

7 Write a program that gets **two** words from the user and then displays a message saying if the first word is an anagram of the second word or not.

For example:

- The word EAT is an anagram of the word TEA as both words contain the same amount of the same letters.
- The word EAT is not an anagram of TEAM as TEAM contains the letter M which is not in EAT.
- The word EAT is not an anagram of AT as EAT contains the letter E which is not in AT.
- The word MEET is not an anagram of MET as although both words only use the letters M, E and T, the first word contains more Es than the second word.

You may assume that the user will only enter words that consist of uppercase letters.

Evidence that you need to provide

(a) Your PROGRAM SOURCE CODE.

(12)

(b) SCREEN CAPTURE(S) showing the result of testing the program by entering:

- the word STOOL followed by the word TOOLS
- the word THE followed by the word HE.

(1)
(Total 13 marks)