EXAM PAPERS PRACTICE

2.2 Queues

Name: ____________________

Class: ____________________

Date: ____________________

Time: **45 minutes**

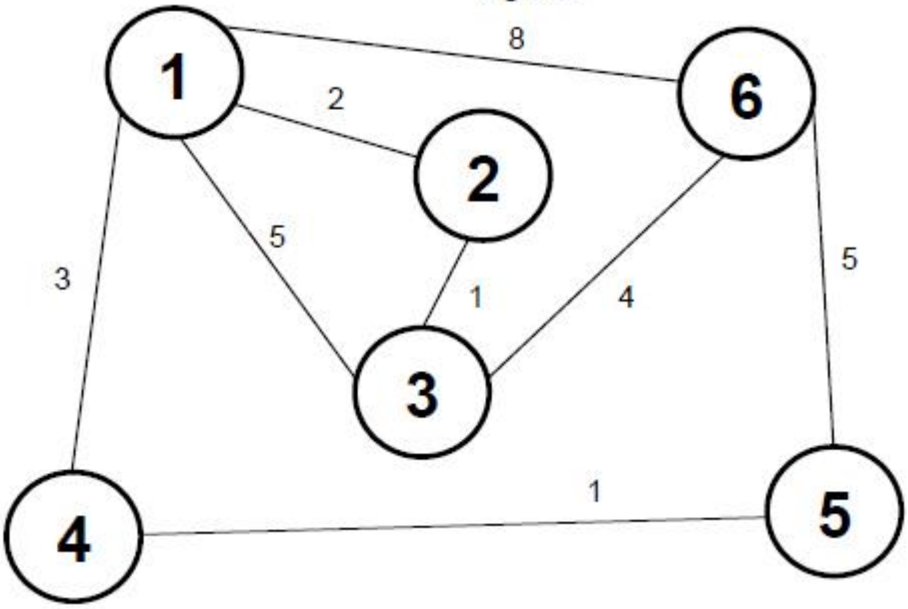
Marks: **39 marks**

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.

(a) State the name of an identifier for a built-in function used in the Skeleton Program that has a string parameter and returns an integer value.

___

___

(1)

(b) State the name of an identifier for a local variable in a method in the `QueueOfTiles` class.

___

___

(1)

(c) The `QueueOfTiles` class implements a linear queue. A circular queue could have been used instead.

Explain why a circular queue is often a better choice of data structure than a linear queue.

___

___

___

___

(2)

(d) It could be argued that the algorithms for a linear queue lead to simpler program code.

State **one** other reason why a linear queue is an appropriate choice in the Skeleton Program even though circular queues are normally a better choice.

___

___

(1)

(e) State **one** additional attribute that must be added to the `QueueOfTiles` class if it were to be implemented as a circular queue instead of as a linear queue.

___

___

(1)

(f) Describe the changes that would need to be made to the Skeleton Program so that the probability of a player getting a one point tile is the same as the probability of them getting a tile worth more than one point. The changes you describe should not result in any changes being made to the points value of any tile.

You should **not** change the Skeleton Program when answering this question.

(4)

(g) The `GetChoice` subroutine uses a built-in function to convert a string to uppercase.

Describe how a string consisting of lowercase characters could be converted to uppercase using only one iterative structure if the programming language being used does not have a built-in function that can do this conversion.

You may assume that only lowercase characters are entered by the user.

You should not change the Skeleton Program when answering this question.

(4)

(Total 14 marks)

Q3.

A computer program is being developed to play a card game on a smartphone. The game uses a standard deck of 52 playing cards, 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`.

The figure below 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.

`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)