Answer **all** the questions.

1. The array queue shown below is set up to hold a small queue. Assume that there is sufficient storage to hold all necessary additions to the queue.

**Queue**


The table below shows variables that are used to maintain the queue:

| Variable | Type | Purpose |
|---|---|---|
| front | integer | pointer to the front element of the queue |
| rear | integer | pointer to the rear element of the queue |
| queue_full | Boolean | indicates whether the queue is full |
| max | integer | the maximum size of the queue |

Shown below is an algorithm that is intended to add an item to the queue.

```
procedure add_to_queue (item)

    if rear==max then
      queue_full=true

    else
       front=front + 1
       queue[front]=item

     endif

endprocedure
```

(i) This algorithm contains a logic mistake. Explain what the mistake is.

---

---

---

**[2]**

(ii) Rewrite the algorithm to correct the mistake.

---

---

---

---

---

**[2]**

2. The layout for a 2-player board game is shown in Fig 2.1

| START | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| END | 62 | 61 | 60 | 59 | 58 | 57 | 56 |

Fig 2.1

The game is played by rolling two 6-sided dice and moving that number of spaces. Both players start on the START space. If a player lands on a space occupied by the other player, they move to the next available space.

The board is to be stored as a 2-dimensional array.

Each time a player moves, a series of obstacles are to be added to the board.

On their turn, each player rolls two dice. The smaller number from the two dice is taken, and that many obstacles will appear on the board in random locations.

For example, if a 3 and 6 are rolled, then 3 obstacles will appear.

A recursive function is written in pseudocode to perform this task.

```
01 function generateObstacle(diceNumber)
02     if diceNumber == 0 then
03         return true
04     else
05         x = randomNumber(0, 7)
06         y = randomNumber(0, 7)
07         board(x, y) = new obstacle()
08         generateObstacle(diceNumber-1)
09     endif
10 endfunction
```

The code `new obstacle()` generates an instance of the object obstacle.

(i) Explain the purpose of the code in line `01` in the algorithm.

---

---

---

---

**[2]**

(ii) Identify the line of code where recursion occurs.

---

**[1]**

(iii) The recursive function could have been written using iteration.

Describe the benefits and drawbacks of using recursion instead of iteration.

Benefits

---

---

---

---

Drawbacks

---

---

---

---

**[4]**

(iv) Rewrite the function so it uses iteration instead of recursion.

---

---

**[4]**

(v) If a position on the board is not occupied, its value is set to a blank string ("").

The current algorithm does not check if the random space generated is currently occupied.

Write a subroutine that takes the generated position of the board, checks if it is free and returns `true` if free, or `false` if occupied.

**[3]**

3(a). A 1-dimensional array stores a set of numbered cards from 0 to 7. An example of this data is shown in Fig in 4.1

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

Fig 4.1

The programmer wants to search for a specific card in the array.

State whether a binary search or a linear search would be the most appropriate method to search for a specific card, and justify your answer.

Search method _____

Justification _____

_____

_____

_____ **[3]**

(b). A programmer is writing a computer program to sort the cards into the correct order (0 to 7).

(i) Show how an insertion sort would sort the array in Fig 4.1 into the correct order. Draw the array after each move.

_____

_____

_____

_____

_____

_____

_____

_____

_____ **[3]**

(ii) Describe how a quick sort algorithm works with the data in Fig 4.2.

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

Fig 4.2

**[6]**

4(a). A salesman travels around the country, stopping at specific places, and then returning to the starting place.

Fig 6.1 shows an example map of places that the salesman visits.

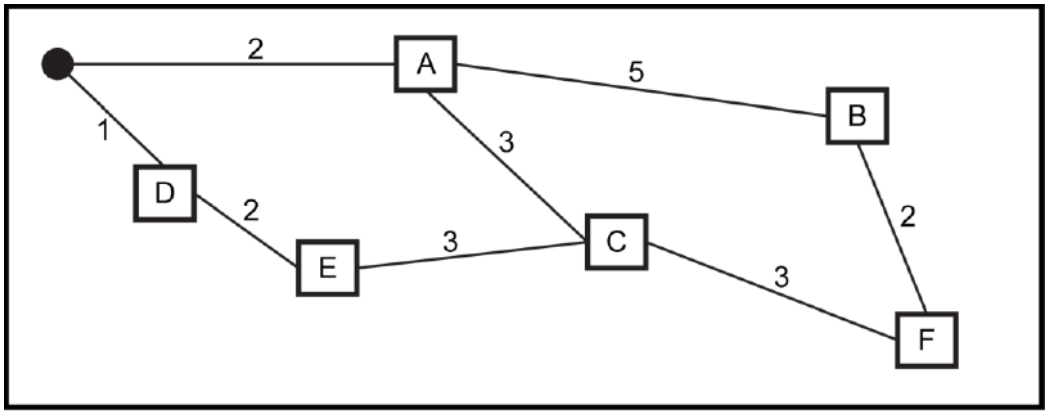


Fig 6.1

The filled in circle represents the start and end point. The letters represent the places to visit. The lines are the routes available and the numbers are the length of time each route takes to travel.

Explain how abstraction has been applied in the production of Fig 6.1

---

---

---

---

**[2]**

(b). The travelling salesman aims to find the shortest route between these places to visit.

A programmer is writing an algorithm to solve the travelling salesman problem.

The programmer is using a tree to find the most efficient route. Fig 6.2 shows part of the tree with three levels completed.

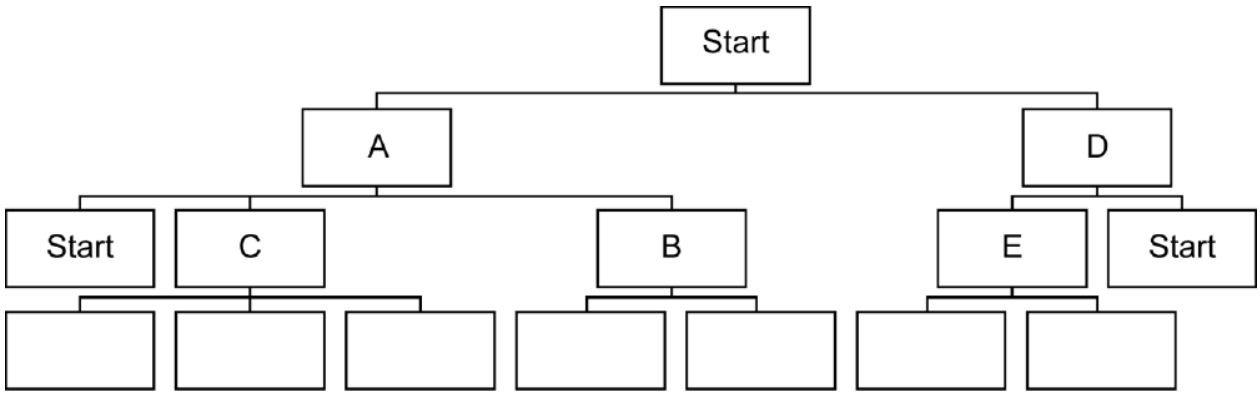


Fig 6.2

(i) The 'Start' nodes on level three are not expanded again as this is a repeat, 'Start' has already been expanded.

Write the place names in the boxes in Fig 6.2, to complete the fourth level of the tree structure for the map shown in Fig 6.1.

**[3]**

(ii) Explain why the tree in Fig 6.2 is **not** a binary tree.

---

---

**[1]**

(c). The programmer has decided to use a graph instead of a tree structure.

(i) Describe what is meant by a graph structure.

---

---

---

---

**[2]**

(ii) The pseudocode below shows part of an algorithm which uses a queue to traverse the graph breadth-first. Complete the missing elements of the algorithm.

```
markAllVertices (notVisited)
createQueue()
start = ..............
markAsVisited(..............)
pushIntoQueue(start)
while QueueIsEmpty() == ..............
    currentNode = removeFromQueue()
    while allNodesVisited() == false
        markAsVisited(..............)
        //following sub-routine pushes all nodes connected to
        //currentNode AND that are unvisited
        pushUnvisitedAdjacents()
    endwhile
endwhile
```

**[4]**

(d). Fig 6.3 is a graph representation of the places that the travelling salesman visits. Using this graph, show how Dijkstra's algorithm would find the shortest path from place A to place F.

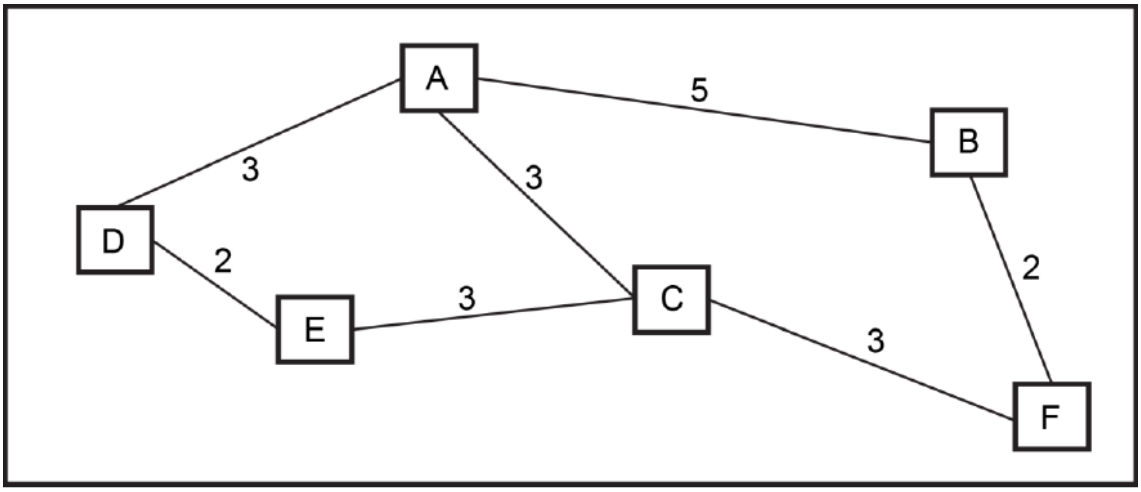


Fig 6.3

---

---

---

---

---

---

---

---

---

---

---

---

---

**[6]**

5(a). Describe an algorithm to insert one data item into a queue data structure.

---

---

---

---

---

---

---

---

**[4]**

(b).

(i) Describe how an insertion sort is performed.

---

---

---

---

---

---

**[3]**

(ii) Demonstrate an insertion sort to place the following numbers into descending numerical order.

12 7 4 5 26

---

---

---

---

---

---

---

---

**[4]**

(iii) State **one disadvantage** of an insertion sort compared with a quick sort.

---

---

**[1]**

6. A programmer decides to use a dynamic data structure to hold items.

Part of the data held is as follows:

(42, 83, 27, 18, 52)

(i) Explain why a binary search would **not** be used for this data.

---

---

---

---

**[2]**

(ii) Describe the steps that a serial search would take to find the value 27.

---

---

---

---

---

---

**[4]**

(iii) Demonstrate the steps needed for a quick sort on these values: (42, 83, 27, 18, 52).

**[5]**

7. An algorithm will make use of a function, contains, that compares two strings and checks if the second string contains the first string. For example, calling the function with ("fox", "foxhound") this would return true.

The function needs to:

- Take two strings as parameters, string1 and string2
- return true if string1 is contained within string2, or both strings are identical
- return false if string1 is not contained within string2

Write, using pseudocode, the function contains.

Annotate your pseudocode with comments to show how it solves the problem.

**[7]**

**END OF QUESTION PAPER**