**Q1.**

The algorithm below shows a bubble sort algorithm represented using pseudo-code. The algorithm sorts the data in a list L.

PROCEDURE BubbleSort(L)

  N ← LEN(L) – 2

  Count1 ← 0

  WHILE Count 1 < LEN(L) – 1

     FOR Count2 ← 0 TO N

       IF L[Count2] > L[Count2 + 1] THEN

         Temp ← L[Count2]

         L[Count2] ← L[Count2 + 1]

         L[Count2 + 1] ← Temp

       ENDIF

     ENDFOR

     Count1 ← Count1 + 1

  ENDWHILE

ENDPROCEDURE

Describe **two** changes that could be made to this bubble sort algorithm that would be likely to result in fewer comparisons being made when sorting the list L. The algorithm should still be a bubble sort algorithm if your suggested changes were made.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(Total 4 marks)**

**Q2.**

How would the infix expression 5 − 3 be represented in Reverse Polish notation?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(Total 1 mark)**

**Q3.**

How would the infix expression 3 + 4 \* 2 − 1 be represented in Reverse Polish notation?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(Total 2 marks)**

**Q4.**

Explain why Reverse Polish notation is sometimes used instead of infix notation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(Total 2 marks)**

**Q5.**

To evaluate an expression in Reverse Polish notation, you start from the left hand side of the expression and look at each item until you find an operator (eg + or −).

This operator is then applied to the two values immediately preceding it in the expression. The result obtained from this process replaces the operator and the two values used to calculate it. This process continues until there is only one value in the expression, which is the final result of the evaluation.

For example 5 2 7 + + would change to 5 9 + after the first replacement.

Explain how a stack could be used in the process of evaluating an expression in Reverse Polish notation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(Total 3 marks)**

**Q6.**

**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.



(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.



**(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)**

**Q7.**

The table below lists some well-known algorithms.

|  |
| --- |
| **Algorithm** |
| Linear search |
| Merge sort |
| Binary search |
| Post-order tree-traversal |

(a)     Which of the algorithms listed in the table has *0*(*n* log *n*) time complexity?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(b)     How many of the algorithms listed in the table are algorithms used to solve tractable problems?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 2 marks)**

**Q8.**

(a)     State the time complexity for the bubble sort algorithm in terms of 𝑛, where 𝑛 is the number of items in the list to be sorted.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(b)     Explain why the bubble sort algorithm has the time complexity stated in your answer to part **(a)**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

**(Total 3 marks)**

**Q9.**

Reverse Polish Notation is an alternative to standard infix notation for writing arithmetic expressions.

(a)     Convert the Reverse Polish Notation expressions in the table to their equivalent infix expressions.

|  |  |
| --- | --- |
| **Reverse Polish Notation** | **Equivalent Infix Expression** |
| 18   9   – |   |
| 10   4   –   12   × |   |

**(2)**

(b)     State **one** advantage of Reverse Polish Notation over infix notation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 3 marks)**

**Q10.**

A binary search tree can be used to represent a list of data so that it can be efficiently searched. **Figure 1** shows an example of a binary search tree:

**Figure 1**

****

(a)     The tree in **Figure 1** is to be searched for data item "Lisa". The tree does not contain "Lisa".

List the data items that will be examined, in the order that they will be visited, when "Lisa" is searched for.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(b)     Tick **one** box in the table to indicate the time complexity of the algorithm used to search for data in a binary search tree.

|  |  |  |
| --- | --- | --- |
|   | **Time Complexity** | **Tick one box** |
|   | O(n) |   |
|   | O(log n) |   |
|   | O(n2) |   |

**(1)**

(c)     In **Figure 2** below, show how the tree in **Figure 1** could be represented by a **Start Index**, together with a one-dimensional array of records, each of which contains the fields **Left Pointer**, **Data** and **Right Pointer**:

**Figure 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|   |   | **Index** | **Left Pointer** | **Data** | **Right Pointer** |
|   |   | [1] |   |   |   |
| **Start Index** |   | [2] |   |   |   |
|   |   | [3] |   |   |   |
| = \_\_\_\_\_\_\_\_ |   | [4] |   |   |   |
|   |   | [5] |   |   |   |
|   |   | [6] |   |   |   |
|   |   | [7] |   |   |   |

**(3)**

(d)     The array shown in **Figure 2** is an example of a static data structure.

Explain the differences between a static data structure and a dynamic data structure, and what the heap is used for with a dynamic data structure.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(3)**

(e)     An in-order traversal is carried out on the binary tree in **Figure 1** to output the values stored in the nodes of the tree.

**Figure 1**

****

(i)      Write out the data items from the tree, in the order that they will be output during the traversal.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(ii)     What is the significance of the order that the data items have been output in?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(f)     Graph traversal is a more complex problem than tree traversal. State **one** feature that a graph might have, which a tree cannot have, that makes graph traversal more complex.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 11 marks)**

**Q11.**

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)**

**Q12.**

Convert the following Reverse Polish Notation expressions to their equivalent infix expressions.

(a)     3 4 \*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(b)     12 8 + 4 \*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(c)     Reverse Polish Notation is an alternative to standard infix notation for writing arithmetic expressions.

State **one** advantage of Reverse Polish Notation over infix notation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 3 marks)**

**Q13.**

A computer program is being developed to allow commuters to plan journeys on the London Underground railway network which connects together over 250 stations.

The program needs to store a representation of the network so that the **shortest route** (ie shortest distance) between any two stations can be found.

**Figure 1** is a map of central London, showing the location of ten of the stations on the London Underground. The locations of the underground railway lines are not shown. Note that nine of the stations are indicated by the symbol  but Charing Cross has a different symbol  because it is a combined underground and overground station.

**Figure 1**

****

**Figure 2** is a map of part of the underground railway network, showing the same ten stations. This map does not show the streets above ground but instead shows the underground railway lines that connect the stations together.

**Figure 2**

|  |
| --- |
|  *Due to copyright restrictions we are unable to show this image. Please use the link below to find the appropriate section of the tube map.*[Standard Tube map - Transport for London](https://www.tfl.gov.uk/cdn/static/cms/documents/standard-tube-map.pdf)  |

**Figure 2** can be used in conjunction with a table of distances between adjacent stations to calculate the shortest route between any two stations on the network.

The map of the entire underground railway network (**not** just the parts shown in **Figure 1** and **Figure 2**) together with the full table of distances can be represented logically as a graph.

(a)     The representation of the underground railway network as a graph is an abstraction.

Explain what an abstraction is.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(b)     Write a detailed description of:

•        how the underground railway network and table of distances could be represented as a graph, **and**,

•        how this representation could be implemented as either an adjacency matrix **or** an adjacency list (describe **one** of these alternatives only), using array(s) in a programming language that does not have a built-in data structure for graphs.

Your implementation should store all the details that are required to calculate the shortest distance between any two stations, but you do not need to describe how the shortest distance would be worked out.

In your answer you will be assessed on your ability to use good English, and to organise your answer clearly in complete sentences, using specialist vocabulary where appropriate.

You may use diagrams to help clarify your description, but as you are being assessed on your ability to use good English, you must ensure that all diagrams are fully explained.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(8)**

**(Total 9 marks)**

**Q14.**

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 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)**

**(Total 5 marks)**

**Q15.**

Reverse Polish Notation is an alternative to standard infix notation for writing arithmetic expressions.

(a)     Convert the following Reverse Polish Notation expressions to their equivalent infix expressions.

|  |  |
| --- | --- |
| **Reverse Polish Notation** | **Equivalent Infix Expression** |
| 45   6 + |  |
| 12   19   +   8   \* |  |

**(2)**

(b)     State **one** advantage of Reverse Polish Notation over infix notation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(c)     The pseudo-code algorithm below can be used to calculate the result of evaluating a Reverse Polish Notation expression that is stored in a string. The algorithm is designed to work only with the single digit denary numbers 0 to 9. It uses procedures and functions listed in the table below, two of which operate on a stack data structure.

StringPos ← 0
Repeat
  StringPos ← StringPos + 1
  Token ← GetCharFromString(InputString, StringPos)
  If Token = ‘+’ Or Token = ‘-’ Or Token = ‘/’ Or Token = ‘\*’
    Then
      Op2 ←Pop()
      Op1 ← Pop()
      Case Token Of
        ‘+’: Result ← Op1 + Op2
        ‘-’: Result ← Op1 - Op2
        ‘/’: Result ← Op1 / Op2
        ‘\*’: Result ←Op1 \* Op2
      EndCase
      Push(Result)
    Else
      IntegerVal ←ConvertToInteger(Token)
      Push(IntegerVal)
  EndIf
Until StringPos = Length(InputString)
Output Result

|  |  |  |
| --- | --- | --- |
| **Procedure/Function** | **Purpose** | **Example(s)** |
| **GetCharFromString**(InputString:String, StringPos:Integer):   Char | Returns the character at position StringPos within the string InputString.Note that the leftmost letter is position 1, not position 0. | GetCharFromString("Computing", 1) would return the character 'C'.GetCharFromString("Computing", 3) would return the character 'm'. |
| **ConvertToInteger**(ACharacter: Char):Integer | Returns the integer equivalent of the character in ACharacter. | ConvertToInteger('4') would return the integer value 4. |
| **Length** (AString: String): Integer | Returns a count of the number of characters in the string AString. | Length("AQA") would return the integer value 3. |
| **Push** (ANumber: Integer) | Puts the number in ANumber onto the stack. | Push(6) would put the number 6 on top of the stack. |
| **Pop** (): Integer | Removes the number from the top of the stack and returns it. | X ← Pop() would remove the value from the top of the stack and put it in X. |

(d)     Complete the table below to trace the execution of the algorithm when InputString is the string: 64+32+\*

In the Stack column, show the contents of the stack once for each iteration of the Repeat..Until loop, as it would be at the end of the iteration.

The first row and the leftmost column of the table have been completed for you.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **StringPos** | **Token** | **IntegerVal** | **Op1** | **Op2** | **Result** | **Stack** |
| 0 | - | - | - | - | - |   |
| 1 |   |   |   |   |   |  |
| 2 |   |   |   |   |   |  |
| 3 |   |   |   |   |   |  |
| 4 |   |   |   |   |   |  |
| 5 |   |   |   |   |   |  |
| 6 |   |   |   |   |   |  |
| 7 |   |   |   |   |   |  |

**(5)**

Final output of algorithm: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(e)     A programmer is going to implement the algorithm above in a programming language that does not provide built-in support for a stack data structure.

The programmer intends to simulate a stack by using a fixed length array of 20 integers named StackArray with indices running from 1 to 20 and an integer variable TopOfStackPointer which will be initialised to 0.

Write a pseudo-code algorithm for the Push operation to push a value stored in the variable ANumber onto the stack.

Your algorithm should cope appropriately with any potential errors that might occur.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(4)**

**(Total 13 marks)**

**Q16.**

(a)     Explain what is meant by an *algorithm.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

(b)     One way of checking that an algorithm is correct is to complete a dry run.

Dry run the algorithm in the figure below by completing the table below.

Assume that x has a value of 7.
The MOD operator calculates the remainder resulting from an integer division.

Answer ← True
FOR Count ← 2 To (x - 1) DO
    Remainder ← x MOD Count
    IF Remainer = 0 THEN
        Answer ← False
    ENDIF
ENDFOR

|  |  |  |
| --- | --- | --- |
| **Answer** | **Count** | **Remainder** |
| True | – | – |
|   | 2 | 1 |
|   |   |   |
|   |   |   |
|   |   |   |
|   |   |   |

**(6)**

(c)     What is the purpose of this algorithm?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 9 marks)**

**Q17.**

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



(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)**

**Q18.**

A binary search tree has the following functions defined:

RootValue(T)  Returns the value stored in the root node of the tree T

LeftChild(T)     Returns the left child (subtree) of the root node of the tree T

RightChild(T)  Returns the right child (subtree) of the root node of the tree T

A recursively-defined procedure P with a tree as a parameter is defined below.

Procedure P(T)

    If RightChild(T) exists

       Then P(RightChild(T))

    Output RootValue(T)

    If LeftChild(T) exists

       Then P(LeftChild(T))

EndProc

(a)     What is meant by a recursively-defined procedure?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(b)     (i)      Complete the table below by dry running the procedure call P(T) for the tree T given below.





**(6)**

(ii)     What does the procedure P describe?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

**(Total 9 marks)**

**Q19.**

A *linear search* and a *binary search* are **two** different methods of searching an ordered list. A given list contains 271 items.

(a)     (i)      What is the maximum number of items accessed when searching for a particular item from the given list using a linear search?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(ii)     Explain your answer.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(b)     (i)      What is the maximum number of items accessed when searching for a particular item from the given list using a binary search?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(ii)     Explain your answer.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(c)     An integer array A contains the following items.

**A**

|  |  |
| --- | --- |
| [1] | 23 |
| [2] | 45 |
| [3] | 16 |
| [4] | 12 |
| [5] | 31 |

(i)      Dry run the following algorithm by completing the trace table.

     For Count1 ← 1 To 4
   For Count2 ← 1 To 4
      If A[Count2] > A[Count2 + 1] Then
     Temp ← A [ Count2]
     A[Count2] ← A[Count2 + 1]
     A[Count2 + 1] ← Temp
      EndIf
   EndFor
EndFor

|  |  |  |  |
| --- | --- | --- | --- |
| Count 1 | Count2 | Temp | A |
| [1] | [2] | [3] | [4] | [5] |
| – | – | – | 23 | 45 | 16 | 12 | 31 |
| 1 | 1 |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |

**(5)**

(ii)     What is the purpose of this algorithm?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(iii)     Suggest **one** way the algorithm could be improved.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 11 marks)**

**Q20.**

The data shown below is a list of surnames of 20 motor car policyholders with the number of claims they have each made in the last five years.

|  |  |  |  |
| --- | --- | --- | --- |
|   | PolicyHolder |   | NoOfClaims |
| 1 | Wilcox | 1 | 1 |
| 2 | Adams | 2 | 0 |
| 3 | Pollard | 3 | 0 |
| 4 | Williams | 4 | 0 |
| 5 | Searle | 5 | 3 |
| 6 | Kelly | 6 | 0 |
| 7 | Lewis | 7 | 1 |
| 8 | Franks | 8 | 5 |
| 9 | Patel | 9 | 1 |
| 10 | Li Che | 10 | 0 |
| ... |   | ... | ... |
| ... |   | ... | ... |
| 19 | Wilkinson | 19 | 3 |
| 20 | Veale | 20 | 0 |

(a)     (i)      The user inputs a policyholder. If the surname is found, the program outputs the number of claims for that policyholder.

         Read(SearchName)
For P := 1 To 20 Do

            If PolicyHolder[P] = SearchName

               Then GoTo 200

     GoTo 300

200 : Write(NoOfClaims[P])

300: End

Give **two** reasons why this is badly designed program code.

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

(ii)     Write declaration statements (in a language with which you are familiar) for the PolicyHolder or NoOfClaims data structure above, and one other variable used in the code above.

The programming language I am using is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

(b)     A new task is to design and write code to establish if there are any policyholders who have made five or more claims. The program will output a ‘yes’ or ‘no’ message only.

Write the code for this new task in a programming language with which you are familiar.

*(Hint: Use a loop structure to initiate the loop, and then end the loop when some condition is met.)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(5)**

**(Total 9 marks)**

**Q21.**

A tree has the following functions defined:

RootValue(T)       Returns the contents of the root node of the tree T

LeftChild(T)          Returns the left child of the root node of the tree T

RightChild(T)       Returns the right child of the root node of the tree T

A recursively-defined procedure P with a tree as a parameter is defined below.

     Procedure P (T)
     If LeftChild(T) exists
          then P(LeftChild(T))
     Output RootValue(T)
     If RightChild(T) exists
          then P(RightChild(T))
EndProc

(a)     What is meant by recursively-defined?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(b)     (i)      Complete the table below by dry running the procedure call P(T) for the tree T given below.



|  |  |  |  |
| --- | --- | --- | --- |
|   Procedure Call |   T |     |   Output |
|   P1 |    |     |     |
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |

**(6)**

(ii)     What does procedure P describe?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

**(Total 9 marks)**

**Q22.**

Write down the comparisons needed to look up *Pascal* using a binary search on the following alphabetically sorted list:

Basic, Fortran, Java, Lisp, Pascal, Prolog, Smalltalk.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(Total 3 marks)**

**Q23.**

Write down the comparisons needed to look up *Newcastle* using a binary search on the following list:

Birmingham, Coventry, Liverpool, Manchester, Newcastle, Sheffield, York.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(Total 3 marks)**

**Q24.**

The algorithm below re-arranges numbers stored in a one-dimensional array called **List**. **Ptr** is an integer variable used as an index (subscript) which identifies elements within **List**. **Temp** is a variable, which is used as a temporary store for numbers from **List**.

     Ptr ← I
While Ptr < 10 Do
  If List [Ptr] > List [Ptr+ 1] Then
     Temp ← List [Ptr]
     List [Ptr] ← List [Ptr+l]
     List [Ptr+l] ← Temp
  Endif
  Ptr ← Ptr+ 1
Endwhile

(a)     Dry-run the algorithm by completing the table below, It is only necessary to show those numbers which change at a particular step.

|  |  |  |
| --- | --- | --- |
|    |   | List |
| Ptr | Temp | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] |
|   |   | 43 | 25 | 37 | 81 | 18 | 70 | 64 | 96 | 52 | 4 |
|  |  |  |  |   |   |   |   |   |   |   |   |
|  |  |   |  |  |   |   |   |   |   |   |   |
|  |   |   |   |   |   |   |   |   |   |   |   |
|  |  |   |   |   |  |  |   |   |   |   |   |
|  |  |   |   |   |   |  |  |   |   |   |   |
|  |  |   |   |   |   |   |  |  |   |   |   |
|  |   |   |   |   |   |   |   |   |   |   |   |
|  |  |   |   |   |   |   |   |   |  |  |   |
|  |  |   |   |   |   |   |   |   |   |  |  |
|  |   |   |   |   |   |   |   |   |   |   |   |

**(7)**

(b)     What will happen when **Ptr**= 10?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(c)     If the whole algorithm is now applied to this rearranged list, what will be the values of:

(i)      List[1] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ii)     List[9] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(iii)     List[10]? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(3)**

**(Total 11 marks)**

**Q25.**

A *binary* search and a *linear* search are two different methods of searching a list.

A given list contains 137 items.

(a)     (i)      What is the maximum number of items accessed when searching for a particular item from the given list using a binary search?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ii)     Explain your answer.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(b)     (i)      What is the maximum number of items accessed when searching for a particular item from the given list using a linear search?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ii)     Explain your answer.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(Total 4 marks)**

**Q26.**

A procedure to process an array of numbers is defined as follows.

     Procedure P(Number)
  Repeat
     X ← StartofArray
     Flag ← False
     Repeat
       If Number(X) > Number (X+ 1)
          Then
            Begin
              Temp ← Number(X)
              Number (X) ← Number (X+ 1)
              Number(X+I) ← Temp
              Flag ← True
            End
          X ← X+l
       Until EndofArray
     Until Flag = False
Endproc

The array number, containing 17, 11, 21, 9, 23, 15, is to be processed by this procedure.

(a)     List the array after the outer Repeat loop has been executed once.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

(b)     What algorithm does the procedure P describe?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(c)     What is the purpose of Flag in this procedure?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 4 marks)**

**Q27.**

The memory of a computer holds an array of records, each of which includes name, address and other information.

(a)     What condition is necessary for the binary search (binary chop) process to work correctly?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

(b)     Describe this process to find the position in the array of the record containing a given name.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(5)**

(c)     Why is this search method normally faster than a linear search?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

**(Total 9 marks)**

**Q28.**

The following section of pseudo-code processes a one-dimensional integer array called *List.* The numbers in *List* are stored in ascending order, and x, *Low, High, Middle* are all integer variables. (The function Int returns the whole number part of its parameter.)

Proc Process(Low, High, x)

     Found ← False
Repeat
     Middle ← Int((Low **+** High)/2)
     If List(Middle) = x
     Then Found ← True
     Else If List(Middle) **>** x
        Then High ← Middle –1
        Else Low ← Middle +1 {List(Middle) <x}
Until Found = True

(a)     Complete the following dry-run table for Process (1, 10, 19), given that the integers in the list are:

2,4, 6, 7, 11, 13, 19, 21, 27, 29

|  |  |  |  |
| --- | --- | --- | --- |
| **Low** | **High** | **Middle** | **Found** |
| 1 | 10 |   |   |
|   |   |   |   |
|   |   |   |   |
|   |   |   |   |

**(7)**

(b)     What type of routine does this pseudo-code define?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 8 marks)**

Mark schemes

**Q1.**

**All marks AO2 (analyse)**

Have a flag variable that is set to True if a swap is made and reset to False at the start of each pass / the outer loop // Have a flag variable that is set to True at the start of each pass to indicate that the list is in order and set to False if a swap is made;

change the outer loop so that it would stop repeating if no swaps have been made;

After the inner loop; subtract 1 from N; // alter inner loop (for) upper limit;

by subtracting Count1 from N;

**[4]**

**Q2.**

**Mark is for AO2 (apply)**

5 3 -

**[1]**

**Q3.**

**All marks AO2 (apply)**

3 4 2 \* + 1 –

**Mark as follows:**

**1 mark** correct order for values and + and – either side of the 1

**1 mark** \* directly after 4 2

**Max 1 if any errors**

**[2]**

**Q4.**

**All marks AO1 (understanding)**

Simpler for a machine / computer to evaluate; **A.** easier **R.** to understand

simpler to code algorithm;

Do not need brackets (to show correct order of evaluation/calculation); **A.** RPN

expressions cannot be ambiguous as **BOD**

Operators appear in the order required for computation;

No need for order of precedence of operators;

No need to backtrack when evaluating;

**Max 2**

**[2]**

**Q5.**

**All marks AO1 (understanding)**

(Starting at LHS of expression) push values/operands onto stack; **R.** if operators are also pushed onto stack

Each time operator reached pop top two values off stack (and apply operator to them);

Add result (of applying operator) to stack;

**Max 2** if any errors **Max 2** if more than one stack used

**Note for examiners:** award 0 marks if description is not about a stack / LIFO structure even if the word “stack” has been used

**[3]**

**Q6.**

(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



**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



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

**Q7.**

(a)  **Mark is for AO1 (knowledge)**

Merge sort;

**1**

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

4;

**1**

**[2]**

**Q8.**

(a)  **Mark is for AO1 (knowledge)**

n2 // O(n2);

**A**. other ways of indicating n2 e.g. n^2

**A**. On2

**1**

(b)  **Marks are for AO1 (understanding)**

In each pass through the list n items will be examined;

There will be (at most) n passes through the list;

**2**

**[3]**

**Q9.**

(a)

|  |  |
| --- | --- |
| **Reverse Polish Notation** | **Equivalent Infix Expression** |
| 18 9 - | 18 - 9**A.** (18 - 9)**R.** 9 – 18 |
| 10 4 - 12 × | (10 - 4) × 12**R.** 10 - 4 × 12**A.** \* for × |

**1 mark** per correct infix expression

**2**

(b)     Simpler/quicker for a machine/computer to evaluate // simpler to code algorithm **A.** Easier as BOD **R.** To understand

Do not need brackets (to show correct order of evaluation/calculation);

**N.E.** Does not use brackets

**T.O.** No brackets so less storage space used

Operators appear in the order required for computation;

No need for order of precedence of operators;

No need to backtrack when evaluating;

**A.** RPN expressions cannot be ambiguous as BOD

**1**

**[3]**

**Q10.**

(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(n2) |   |

**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]**

**Q11.**

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

It contains a cycle / cycles;

**1**

(b)     **All marks AO2 (apply)**

|  |  |
| --- | --- |
| **Vertex (inFigure 1** | **Adjacentvertices** |
| **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]**

**Q12.**

(a)     **All marks AO2 (apply)**

3 \* 4

**1**

(b)     **All marks AO2 (apply)**

(12 + 8) \* 4;

**1**

(c)     **Mark for AO1 (understanding)**

**1 mark:** Simpler / easier for a machine / computer to evaluate / / simpler / easier to code algorithm

**R** Simpler / easier to understand

Do not need brackets (to show correct order of evaluation / calculation);

Operators appear in the order required for computation;

No need for order of precedence of operators;

No need to backtrack when evaluating;

**A** RPN expressions cannot be ambiguous as Benefit Of Doubt (BOD)

**1**

**[3]**

**Q13.**

(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]**

**Q14.**

(a)     18, 23, 21, 36, 40, 45, 58, 59

**Mark as follows:**18 in the first place;
23 and 21 in correct order and in the second and third places;
21 and 36 in the correct order and in the third and fourth places;
40, 45, 58 and 59 in the correct order and in the last four places;

**A** Table 3 instead of Table 2 as long as the bottom cell of each of the scores column is correct (**I.** any working out)

**4**

(b)     Bubble sort;
**NE** sort

**1**

**[5]**

**Q15.**

(a)

|  |  |
| --- | --- |
| **Reverse Polish Notation** | **Equivalent Infix Expression** |
| 45  6  + | *45 + 6****R*** *6 + 45* |
| 12 19 + 8 \* | *(12 + 19) \* 8****R*** *12+19\*8, (19+12)\*8****A*** *x for \** |

**2**

(b)     Simpler for a machine / computer to evaluate // simpler to code algorithm
**A** easier **R** to understand
Do not need brackets (to show correct order of evaluation/calculation);
Operators appear in the order required for computation;
No need for order of precedence of operators;
No need to backtrack when evaluating;
**A** RPN expressions cannot be ambiguous as **BOD**

**1**

(c)



Output : **50***1 mark for each of rows 1–3
1 mark for rows 4 and 5 together
1 mark for rows 6 and 7 together
1 mark for correct final output*

*Values of Op1 and Op2 MUST be assigned in rows 3, 6 and 7 to award the marks for these rows. They cannot be inferred from incorrectly entered previous values.*

**I** values in empty cells, even if they are incorrect.

**6**

(d)     If StackArray is full
  Then Stack Full Error
  Else
    Increment TopOfStackPointer
    StackArray [TopOfStackPointer]  
        ANumber
EndIf

*1 mark for appropriate If structure including condition (does not need both Then and Else) – Do not award this mark if ANumber is put into StackArray outside the If.
1 mark for reporting error in correct place
1 mark\* for incrementing TopOfStackPointer
1 mark\* for storing value in ANumber into correct position in array
\* = if the store instruction is given before the increment instruction OR
the If structure then award* ***Max 1*** *of these two marks UNLESS the item is inserted at position TopOfStackPointer+1 so the code would work.*

**I** initialisation of TopOfStackPointer to 0

**A** TopOfStackPointer=20/>=20 for Stack is full
**A** Logic of if structure reversed i.e. If stack is not full / TopOfStackPointer<20 / <>20/!=20 and Then, Else swapped
**A** Any type of brackets or reasonable notation for the array index
**DPT**   If candidate has used a different name any variable then do not award first mark but award subsequent marks as if correct name used.
Refer answers where candidate has used a loop to find position to insert item into stack to team leaders.

**4**

**[13]**

**Q16.**

(a)     *Must have the concept of problem/task for the first mark*

A (step-by-step) description of how to complete a task / a description of a process that achieves some task / a sequence of steps that solve a problem / A sequence of unambiguous instructions for solving a problem;
**R** Set of instructions

Independent of any programming language;
That can be completed in finite time;

**Max 2**

(b)



**Mark as follows:**answer column;
**A** True instead of blank cells
**R** if no evidence that dry run has been attempted
count column;

*1 mark per correct value in remainder column;;;;*

**6**

(c)     Works out if x is a prime number //
Checks if x is divisible (with no remainder);

**1**

**[9]**

**Q17.**

(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]**

**Q18.**

(a)     A procedure/routine that calls itself/ is defined in terms of itself;

**A** Function instead of procedure

**R** re-entrant  **R** program **R** iteration

**1**

(b)     (i)



**6**

(ii)     Reverse Inorder// Reverse order; (tree) traversal;

**2**

**[9]**

**Q19.**

(a)     (i)      271;

**1**

(ii)     The required item might be the 271st one/last one/ not be present// Every item accessed;

**1**

(b)     (i)      9;

**1**

(ii)     Each comparison halves the number of items to be accessed//271 lies between 28 and 29;

**1**

(c)



*1 mark for Count1*

*1 mark for Count2*

*1 mark for Temp*

**5**

(ii)     (bubble) sort the items into ascending order;

**1**

(iii)     Reduce the number of tests each pass// stop when no swaps occur during a pass//Add a flag No Swaps to indicate when no swaps occur// change loop control to Repeat until no swaps// sort variable sized array;

**1**

**[11]**

**Q20.**

(a)     (i)      •        poorly structured code;

•        uses GoTo statements;

•        the flow of control jumps out of a loop;

•        nothing reported to the user when no matching name found;

•        abbreviated variable for ‘position’ variable;

•        ReadLn is better than Read;

•        Program only iterates once / considers only the first array element;

•        (if duplicates) only the first matching surname is found;

•        (loop terminates at 20) does not allow for additional array /name entries;

**A** poor layout - excessive indentation used;

**I.** variable declaration // reference to the syntax

**Max 2**

(ii)     All statements must have correct identifier name correct data type (String / Text // Integer / Byte / Word / Int / Shortint / Short as appropriate)

In addition, either array must have brackets to indicate an ‘array’ 19/20 to indicate a range;

**Max 2**

(b)     Intialisation of counter or Boolean variable

P := 1 / P := 0 / For P := 1 to 20 // IsFound := False;

Looping

LOOP UNTIL // DO WHILE // WHILE DO // REPEAT UNTIL and used at the beginning/end of a code block as appropriate;

Some loop condition is met

(P = 20/21) OR IsFound = TRUE / P = 20/21 // IsFound = TRUE / IsFound;

IF with use of the array

IF NoOfClaims [P];

Selection condition

>4 / >=5;

Loop counter incremented

P = P+1

Final output

Correct logic followed with OUTPUT ‘Yes’

**A** multiple times

Final output

Correct logic followed with OUTPUT ‘No’

**R** Multiple times

**R** ‘Prose’ scores 0

**5**

**[9]**

**Q21.**

(a)     A procedure/routine that calls itself/ is defined in terms of itself;

**A** Function instead of procedure

**R** re-entrant
**R** program
**R** iteration

**1**

(b)     (i)



**6**

(ii)     In order; (tree) traversal

**2**

**[9]**

**Q22.**

Compare Pascal with middle item of list / Lisp;

Compare Pascal with middle item of upper sublist / Prolog;

Compare Pascal with Pascal // compare only item in this sublist to get a match;

*Lose 1 mark if Pascal not explicit in comparison*

*Stop marking from time it goes wrong*

**OR**

List[4] = Pascal? False;
**A** [4] = Pascal
**R** 4 = Pascal

List[6] = Pascal? False;

List[5] = Pascal? True;

*If formula explicit, follow through on formula*

**[3]**

**Q23.**

Compare Newcastle with (middle item of list), Manchester;

Compare Newcastle with (middle item of upper sublist), Sheffield;

Compare Newcastle with Newcastle // compare only item (in lower sublist of this upper sublist) to get a match;

*Lose 1 mark if Newcastle not explicit in comparison stop marking from time it goes wrong*

***OR***

List[4] = Newcastle? False;
**A** [4] = Newcastle
**R** 4 = Newcastle

List[6] = Newcastle? False;

List[5] = Newcastle? True;

*If formula explicit, follow through on formula*

**[3]**

**Q24.**

(a)

|  |  |  |
| --- | --- | --- |
|   |   | List |
| Ptr | Temp | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] |
|   |   | 43 | 25 | 37 | 81 | 18 | 70 | 64 | 96 | 52 | 4 |
| 1 | 43 | 25 | 43 |   |   |   |   |   |   |   |   |
| 2 | 43 |   | 37 | 43 |   |   |   |   |   |   |   |
| 3 |   |   |   |   |   |   |   |   |   |   |   |
| 4 | 81 |   |   |   | 18 | 81 |   |   |   |   |   |
| 5 | 81 |   |   |   |   | 70 | 81 |   |   |   |   |
| 6 | 81 |   |   |   |   |   | 64 | 81 |   |   |   |
| 7 |   |   |   |   |   |   |   |   |   |   |   |
| 8 | 96 |   |   |   |   |   |   |   | 52 | 96 |   |
| 9 | 96 |   |   |   |   |   |   |   |   | 4 | 96 |
| 10 |   |   |   |   |   |   |   |   |   |   |   |

*Ignore Ptr & Temp column*

*1 mark for each of rows 1, 2, 4, 5, 6, 8, 9*

*(Final list 25, 37, 43, 18, 70, 64, 81, 52, 4, 96)*

**7**

(b)     Control will pass to the instruction after Endwhile;

/the instruction/command/statement after Endwhile will be executed;

Program will exit while-block; loop stops;

**A** algorithim stops; **R** program stops;

**Max 1**

(c)     (i)      25;

*If part (a) not fully correct allow follow through: or lower of [1] & [2]*

**3**

(ii)     81;

*Only allow follow through mark if the list at the end of part(a) is still a partially sorted list*

(iii)     96;

*Must be 96 in all cases*

**[11]**

**Q25.**

(a)     (i)      8;

**1**

(ii)     Each time a comparison is made in a binary search the number of items to be searched / list is halved;

// 137 lies between 27 and 28;

Could give (ii) even if (i) incorrect

**1**

(b)     (i)      137;

**1**

(ii)     In a linear search of 137 items, the required item might be the 137th one;
*Need a termination – must explain why 137 is the maximum*

**1**

**[4]**

**Q26.**

(a)     11, 17, 9,21,15,23;

*(2 if all right, 1 if 4 of 6)*

*If > misinterpreted, follow through for 1 mark*

**2**

(b)     A bubble sort;

**1**

(c)     To detect when all the numbers have been sorted

Efficiency (to stop procedure repeating unnecessarily);

**R** to detect when numbers have switched

**1**

**[4]**

**Q27.**

(a)     Array must be sorted (1), on the field being used as the search key (1)

**2**

(b)     Description must include the following points: Find median record of array **(l)** Compare key field of record at median position with required search key, exit if found (1)If search key lower (i.e. required record in first half), discard second half, else discard first half (1)Repeat process (1) until either found, or no further division possible so record does not exist (1)

**5**

(c)     On each iteration, half the possible matches are eliminated, compared with only one for the linear search (2)

Linear search on average scans n/2 records, compared with log2n which is smaller “Looks at fewer records” without further explanation (1)

**2**

**[9]**

**Q28.**

(a)

|  |  |  |  |
| --- | --- | --- | --- |
| Low | High | Middle | Found |
|  |  | *5* |   |
| *6* |  | *8* |   |
|  | *7* | *6* |   |
| *7* |  | *7* | true |

*1 mark for each entry above (as far as first incorrect entry)*

*Mark row by row*

**Max 7**

(b)     Binary search/chop

Iterative (no synonyms)

(Specific searches not on AS syllabus - search sufficient for mark)

**1**

**[8]**