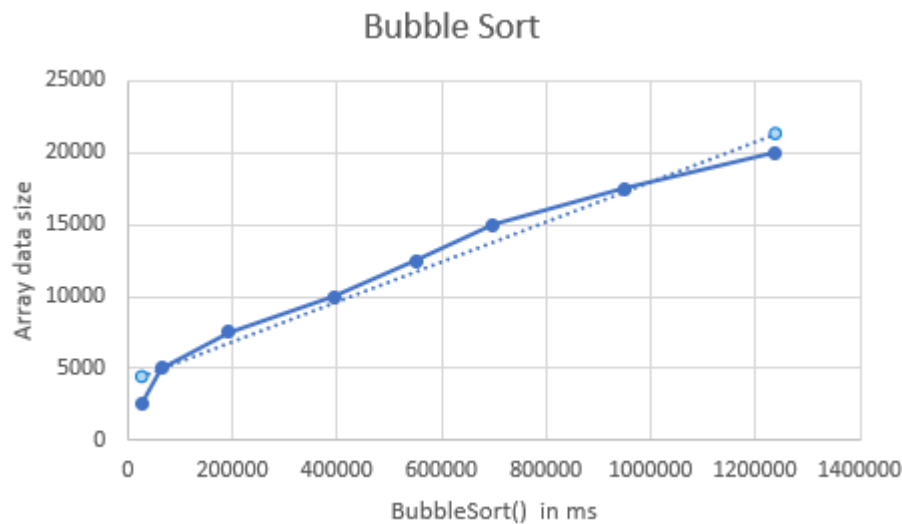


## Coding Exam

1)

a)

**Q1.a****Bubblesort Data**

Suitable Chart

Correctly Labelled Y axis

Correctly Labelled X axis

Trend-line

**Quicksort Data**

Suitable Chart

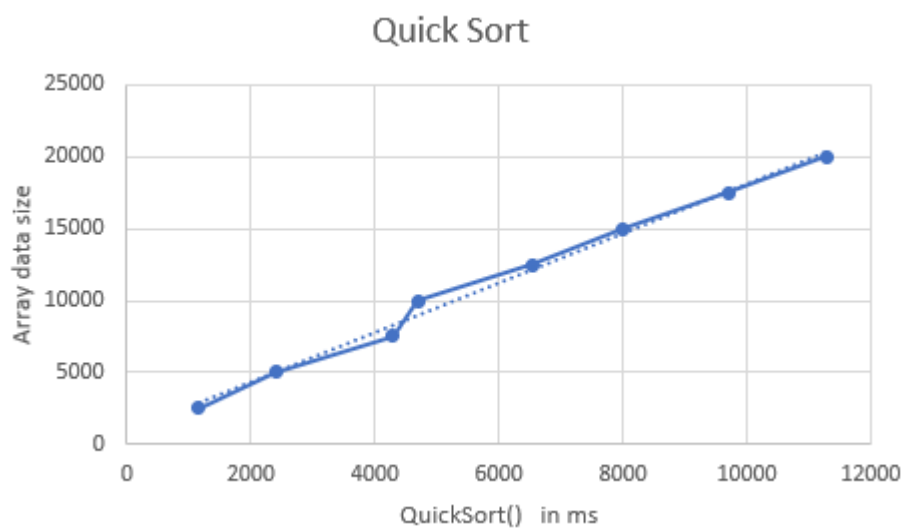
Correctly Labelled Y axis

Correctly Labelled X axis

Trend-line

**Estimated Big O**

Not answered

**[Marks 8 from 10]**

b)

1.b - Not answer given

c)

i. recursive algorithms are problems that require solutions that are based on smaller versions of the problem

ii.

Q1.c.i - Very brief answer with some omissions [2/5]

Q1.c.ii - Recursive code incorrectly identified [0/2]

```

WHILE NOT done:
  WHILE left <= right AND array[left] <= pivot
    SET left TO left + 1
  END WHILE
  WHILE array[right] >= pivot AND right >= left
    SET right TO right -1
  END WHILE
  IF right < left THEN
    SET done TO True
  ELSE
    SET temp TO array[left]
    SET array[left] TO array[right]
    SET array[right] TO temp
  END IF
END WHILE

```

This is an example of a recursion

2)

a)

Depth first search algorithms operate by making sure that all nodes on one depth are searched before going lower. This can be seen by Figure 2.1 as it starts at the root node 'A', goes left to 'B' and back up and around to 'C'. This shows that depth first algorithms are thorough as they make sure they don't miss any information. This can make it inefficient if processing lots of data.

As stated earlier, this algorithm will take longer than others, specifically if required to find specific paths or the 'shortest path' to a node. Generally, the algorithm will require less memory than a breadth first algorithm.

Q2.a - Depth First will start at the root (A) and traverse down a complete branch until it reaches the leaf node, before moving to the next branch. The answer given describes a Breadth First search. The analysis of the algorithm given is very brief with some omissions [2/9]

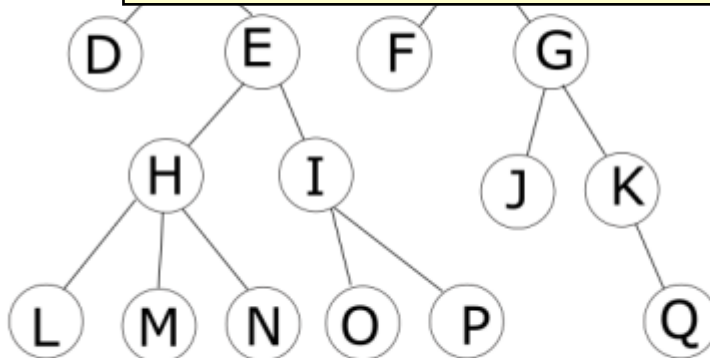
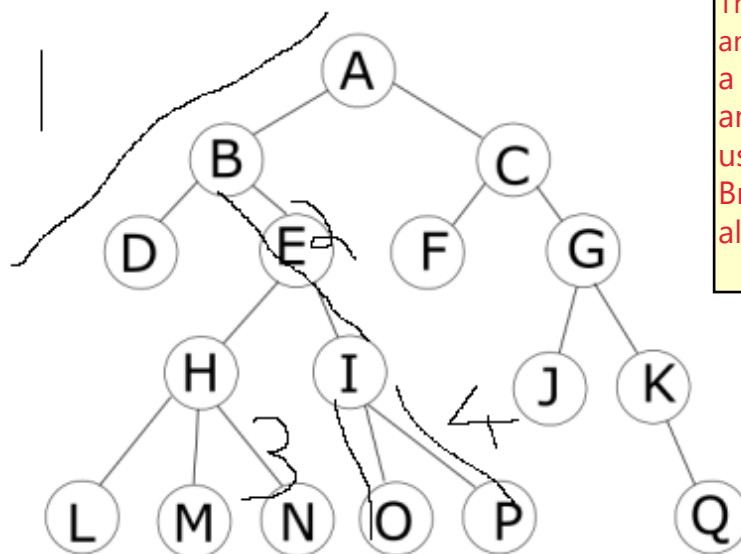


**Figure 2.1**

If the algorithm needed to find a way from the root node to P it would operate like this



Q2.b

The answer is incomplete and doesn't describe how a queue data structure and while loop can be used to implement a Breadth First Search algorithm. [0/10]

**Figure 2.1**

The path would end up being A, B, D, E, I, O, P. This can be less efficient but it will get it done.

c)

queue's operate as first in first out systems, so the data that goes in will come out the same order it went in. the main operations are add() and remove(). Add() adds data to the list, remove() removes it.

Q2.c

The description is generally correct; but a little brief and uses the incorrect terminology i.e. push() to add, pop() to remove. [4/6]

3

a)

i.

77, 40, 39, 42, 86, 85, 90

ii.

39, 42, 40, 85, 90, 86, 77

iii.

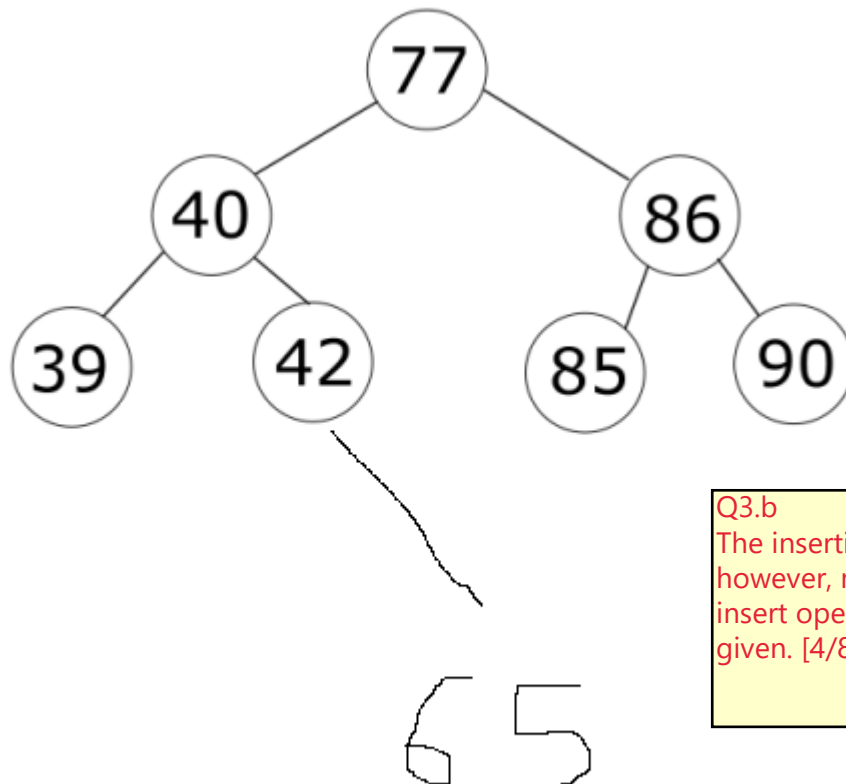
39, 40, 42, 77, 85, 86, 90

b)

65 would go into binary search tree below 42. This is because it gets put as a node on the number it is closest to. It would look something like this

Q3.a

The order of nodes correctly identified for each traversal method. [12/12]

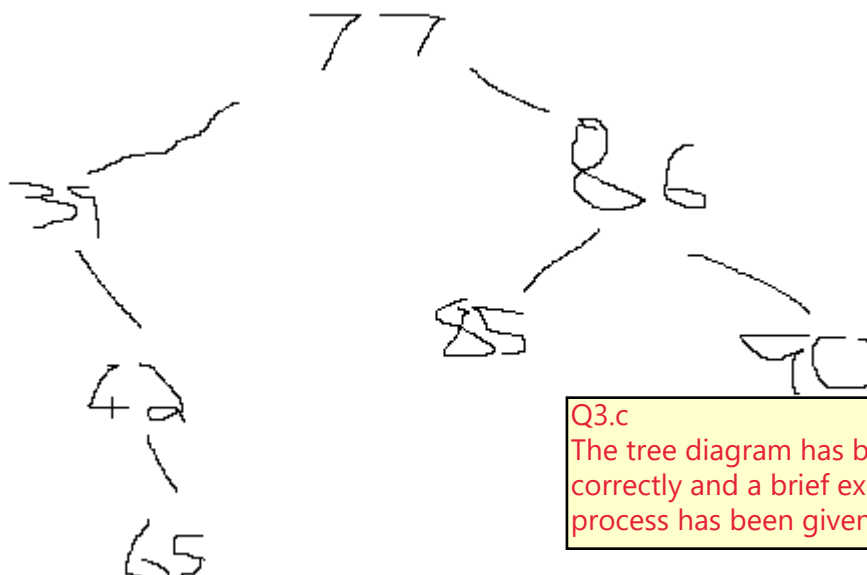


Q3.b

The insertion point is correct, however, no analysis of the insert operation has been given. [4/8]

c)

when 40 is deleted the number closest to 40 replaces it. In this case it is 39.



Q3.c

The tree diagram has been reproduced correctly and a brief explanation of the process has been given. [4/5]

This is how the search tree would look when 40 is replaced

4) Question 4 wasn't required as the exam instructions clearly stated -

a) **SECTION B - ANSWER TWO OUT OF THE THREE QUESTIONS IN THIS SECTION.**

a has functions. To reference record in a table, users would use the get function for instance `get(name)` and then with a simple `cout "name";` the name would be outputted.

b)

closed hashing works by making sure no data collides with other bits of data. it does this by forcing them to find alternate locations. I believe 36 will be stored in row 5 as  $\text{Index} = (\text{hash}(v) + i)$  just means that they are being added to the array and considering the array starts at 3 and there are 2 other results, 5 would be the logical place to put it.

|    |    |
|----|----|
| 0  |    |
| 1  |    |
| 2  |    |
| 3  | 8  |
| 4  | 23 |
| 5  | 36 |
| 6  |    |
| 7  |    |
| 8  |    |
| 9  |    |
| 10 |    |

**Table 4.1**

c) if quadratic probing is used, 36 would end up in row 9. it works by squaring the successive numbers and considering the function used to insert squares  $i$  which I believe to be  $3 \times 3$  is 9.

|    |    |
|----|----|
| 0  |    |
| 1  |    |
| 2  |    |
| 3  | 8  |
| 4  | 23 |
| 5  |    |
| 6  |    |
| 7  |    |
| 8  |    |
| 9  | 36 |
| 10 |    |

**Table 4.1**

it would solve the issue of clustering as none of the data would be stored close in the table. If my calculations are correct the next entry would go into slot 19 as  $4 \times 4$  is 16+3 is 19.

d)

separate chaining is another way of preventing collisions, it is also easier to implement than other solutions. It allows data to be hashed to the same keys. To do this it facilitates the creation of long link chains. They can be long, thus any number of bits of data can be added to this key.

This however does mean that there are wasted keys within the table as not all data will be allocated to a separate key.

**Summary:**

Question 1: 10

Question 2: 7

Question 3: 20

Question 4:

Total: 37 from 75  
(49%)