

Advanced Software Development Exam Answers:

Section A:

Question 1:

A:

Q1.a - No answer given

B:

The bubble sort algorithm has a time complexity of $O(n^2)$ because no matter how much data you put in; it will always require the same amount of steps as data each time you add data to it.

The Quick Sort – Partition algorithm has a time complexity of $O(n \log n)$ because the array is divided into two parts and each time it takes $O(n)$ meaning it is $O(n \log n)$ because the time goes up linearly as the n goes up exponentially.

The Quick Sort Algorithm has a time complexity of $O(n \log n)$ because the array is divided into two parts and each time it takes $O(n)$ because the time goes up linearly as the n goes up exponentially.

Q1.b

The response is mostly correct for quick sort; however, the analysis is incorrect for the bubblesort algorithm.

[Marks 4 from 8]

C:

Recursive algorithms normally call themselves with small input values and will return the result for the input by carrying out some basic tasks on the returned value for the small input. In the quick sort algorithm shown in the exam paper, recursion is shown during the following lines:

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

Q1.c

Correct explanation of recursion given but a little brief.

Recursive code incorrectly identified

[Marks 3 from 7]

It takes the input value . This is what a recursive algorithm does. It uses the small input value and performs the quicksort operation on the small value thus making it +1.

Section B:

Question 3:

A:

Using the binary tree that has been shown in figure 3.1, the nodes can be visited in three different orders. Pre-order traversal, post-order traversal and in-order traversal.

For the pre-order traversal, the order is as follows: 77, 40, 39, 42, 86, 85 and 90. The first node visited is 77 because it starts from the root node and goes through the tree left to right.

For the post-order traversal, the order is as follows: 39, 42, 40, 85, 90, 86 and 77. The first node visited is 39 because the tree starts on the left, moves to the right and then ends with the root node.

For the in-order traversal, the order is as follows: 39, 40, 42, 77, 85, 86 and 90. The first node visited is 39 because the tree starts from the left, moves to the root node and then goes to the right.

B:

If number 65 was added into the binary search tree that is seen in figure 3.1, it would be added to the right of the number 39 node. The traversals would look like this:

Pre-order: 77, 40, 39, 65, 42, 86, 85 and 90.

Post-order: 39, 42, 65, 40, 85, 90, 86, and 77.

In-order: 39, 40, 65, 42, 77, 85, 86 and 90.

C:

Using the binary tree shown in figure 3.1, removing the number 40 node from the tree will result in the number 39 node becoming the parent for the number 42 node. This is because the number 40 node was the parent node for both the left child (number 39 node) and the right child (number 42 node). Since the parent node (number 40 node) has been removed from the binary tree, the number 39 node will become the new parent and it will have one child which is the number 42 node. Below is an image created to demonstrate what the binary tree would look like if the number 40 node was deleted from it.

Q3.a

The order of nodes correctly identified for each traversal method.

[Marks 12 from 12]

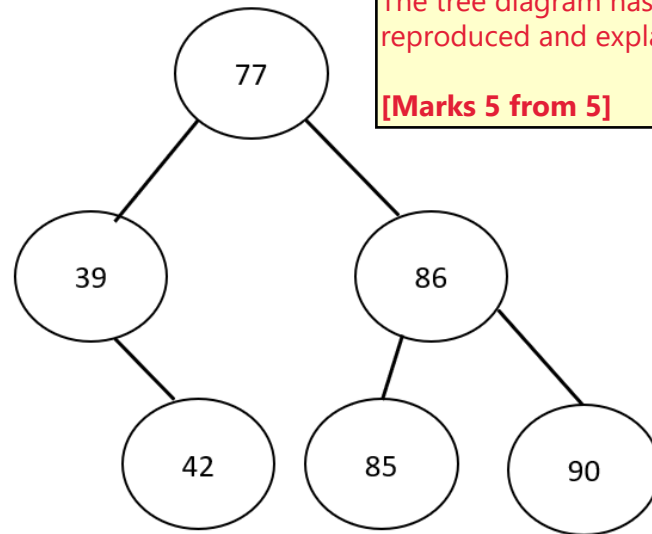
Q3.b

The answer given is incorrect. The process is

65 < 77 (Root) - Left Tree

65 > 40 - Go right - node not null (42)

65 > 42 - Go right - insert new node

**Q3.c**

The tree diagram has been correctly reproduced and explained

[Marks 5 from 5]**Question 4:****A:****Q4.a - Good answer****[Marks 4 from 4]**

Hashing is a process that transforms a key or a string of characters into a different value. A hash table is made up of two columns, an index and data that is stored. Hashing uses hash functions and or hash algorithms to generate the values in the index column. If the data that is stored in the hash table is not numerical, the hash generates a numerical value to assign the data to and then generates the index number to store the data. Essentially, hashing is the process of generating an address to store data in or read data from.

B:

The key value of 36 would be placed into index number 0. This is because 36 has a hash value of 0 when dealing with the hashing table 4.1. and the formula $\text{Index} = (\text{hash}(v) + i) \bmod m$ where $\text{hash}(v) = v \bmod m$.

Q4.b

Incorrect implementation of the Hash algorithm.
Correct calculation would be:

- Hash: $36 \bmod 11 = 3$
- Index = $(3 + 0) \bmod 11 = 3$
- Address 3 not available $i = i + 1$
- Index = $(3 + 1) \bmod 11 = 4$
- Address 4 not available $i = i + 1$
- Index = $(3 + 2) \bmod 11 = 5$
- Address 5 is available - insert 36 at 5

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

Here is what the table looks like with 36 added into it.

C:

The key value of 36 would be placed into index number 3 if quadratic probing was used for Question B. This is because $36 = 3 \times 11 + 3$, therefore 8 would go into 8 and 23 would go into 1.

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

Q4.b

Incorrect implementation of the Hash algorithm. Correct calculation would be:

- Hash: $36 \bmod 11 = 3$
- Index = $(3 + 0^2) \bmod 11 = 3$
- Address 3 not available $i = i + 1$
- Index = $(3 + 1^2) \bmod 11 = 4$
- Address 4 not available $i = i + 1$
- Index = $(3 + 2^2) \bmod 11 = 7$
- Address 7 is available - insert 36 at 7

Here is what the table looks like quadratic probing applied to it.

D:

Separate chaining is a technique that is used to resolve collision of key values in the hash table. Separate chaining is implemented using linked lists and is combined with a hash table to stop collision of key values. The advantages of separate chaining is the fact that it is very easy to implement since it used linked lists and the fact that you can have an unlimited hash table capacity. The

disadvantages of separate chaining are having it be memory inefficient because it requires a secondary data structure and having long chains will produce linear search times.

Q4.c

Good answer

[Marks 6 from 7]

Summary:

Question 1: 7

Question 2:

Question 3: 17

Question 4: 10

Total: 34 from 75 (45%)