Started on	Friday, 25 April 2025, 16:21
State	Finished
Completed on	Friday, 25 April 2025, 16:59
Time taken	38 mins 42 secs
Grade	**163.43** out of 200.00 (**81.71**%)

Question 1

Partially correct

Mark 12.00 out of 40.00

A Binary Search Tree (BST) was created by inserting these integers in the following sequence: 10, 2, 12, 5, 7, 13, 1, 6, 9, 4 (i.e. "10" gets inserted first and "4" inserted last).

Drag-and-drop the correct sequence of integers when traversing the tree using **pre-order depth first traversal**. Note that your sequence must absolutely match the index numbers to the left-most column of the table otherwise marks will be deducted for each incorrect match.

	Correct Integer Sequence
Index 0	10 ✔
Index 1	2 ✔
Index 2	5 ✖ *[1]*
Index 3	6 ✖ *[5]*
Index 4	9 ✖ *[4]*
Index 5	7 ✔
Index 6	4 ✖ *[6]*
Index 7	12 ✖ *[9]*
Index 8	13 ✖ *[12]*
Index 9	1 ✖ *[13]*

12 11 13 8 5 0 2 9 3 10 6 1 7 null 4

Your answer is partially correct.

Question 2

Correct

Mark 50.00 out of 50.00

Complete the ArrayList program segment by dragging-and-dropping the correct arguments, instructions and/or comments onto the blanks below.

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(5);
list.add(3);
list.add(10);
list.add(7);

System.out.println(list.size()); //prints the value [ 4 ]

[ int ] val1 = list.set(2, [ 34 ] );
System.out.println([ val1 ] ); //prints the value [ 10 ]

int val2 = list.[ get ] (2);
System.out.println(val2); //prints the value 34

int val3 = list.remove( [ 3 ] );
System.out.println(val3); //prints the value [ 7 ]

list.remove(1);
System.out.println([ list ] .size()); //prints the value [ 2 ]
```

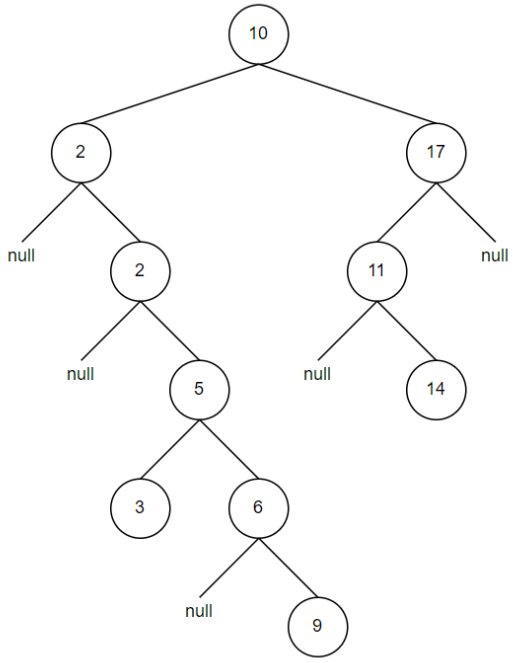
Your answer is correct.

Question 3

Correct

Mark 50.00 out of 50.00

A Binary Search Tree (BST) was created as shown below.



Drag-and-drop the correct sequence of integers when traversing the tree using **post-order depth first traversal**. Note that your sequence must absolutely match the index numbers to the left-most column of the table otherwise marks will be deducted for each incorrect match.

	Correct Integer Sequence
Index 0	3 ✔
Index 1	9 ✔
Index 2	6 ✔
Index 3	5 ✔
Index 4	2 ✔
Index 5	2 ✔
Index 6	14 ✔
Index 7	11 ✔
Index 8	17 ✔
Index 9	10 ✔

17 11 9 null 1 8 15 2 16 10 13 0 5 6 14 7 4 12 3

Question 4

Partially correct

Mark 51.43 out of 60.00

Drag-and-drop the correct sequence of a stack-based solution to reconstruct the Binary Search Tree (BST) based on the **pre-order depth first traversal**, ensuring the left child node is first stored followed by the right child node. Note that your sequence must absolutely match the step numbers to the left-most column of the table otherwise marks will be deducted for each incorrect match.

HINT: The first step is to ensure you have an empty stack.

Step 1	Flush the stack to clean any r emaining data stored on it. ✔	Set this value as left child if it is less than or equal to previous top node's value.
Step 2	Get first value from list then make this as root node by pushing the value and node position onto stack. ✔	Get first value from list then make this as root node by pushing the value and node position onto stack.
Step 3	Get next value from list. ✔	Get next value from list.
Step 4	Set this value as left child if it is less than or equal to previous top node's value. ✔	Set this value as right child if it is less than previous top node's value.
Step 5	Otherwise set this value as right child. ✔	Otherwise set this value as left child.
Step 6	Push this value and its nod e position, i.e. left or right c hild, onto stack. ✔	Repeat steps 2-6 for remaining values in the list.
Step 7	Repeat steps 2-6 for remaining values in the list. ✖	Otherwise set this value as right child.
		Push this value and its node position, i.e. left or right child, onto stack.
		Push this value onto stack.
		Set last value in list as root node then push onto stack.
		Repeat steps 3-6 for remaining values in the list.
		Flush the stack to clean any remaining data stored on it.

Your answer is partially correct.

You have correctly selected 6.
The correct answer is:

Step 1	Flush the stack to clean any remaining data stored on it.
Step 2	Get first value from list then make this as root node by pushing the value and node position onto stack.
Step 3	Get next value from list.

Step 4	Set this value as left child if it is less than or equal to previous top node's value.
Step 5	Otherwise set this value as right child.
Step 6	Push this value and its node position, i.e. left or right child, on to stack.
Step 7	Repeat steps 3-6 for remaining values in the list.