

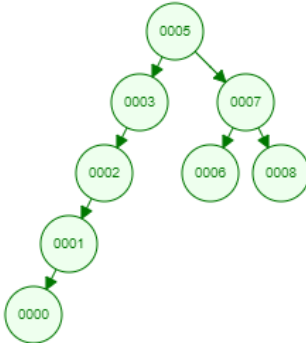


THE NATIONAL INSTITUTE OF ENGINEERING COMPUTER SCIENCE & ENGINEERING

Course Name: **Data Structures**

Course Code: **CS3C01**

Scheme & Solutions

Semester: 3		Date: 18-11-2019	Max. Marks: 25									
Q. No	Questions			Mark s								
1.	<div>Create binary search tree(BST) for input in the order: 5,3,7,2,1,6,8,0. Write i) Preorder ii) Inorder iii) Postorder traversal outputs of this BST. BST: 2 M</div> <div><div>i) Preorder : 5,3,2,1,0,7,6,8</div><div>ii) Inorder : 0,1,2,3,5,6,7,8</div><div>iii) Postorder: 0,1,2,3,6,8,7,5</div></div> <div></div>			5M								
2.	<div>Write recursive algorithms for finding i) Height of binary tree ii) Total nodes in the binary tree.</div> <div><div><div>i) Height (TREE) 3M</div><div>Step 1: IF TREE = NULL Return 0 ELSE SET LeftHeight = Height(TREE->LEFT) SET RightHeight = Height(TREE->RIGHT) IF LeftHeight > RightHeight Return LeftHeight + 1 ELSE Return RightHeight + 1 [END OF IF] [END OF IF] Step 2: END</div></div><div><div>ii) TotalNodes(TREE) 2M</div><div>Step 1: IF TREE = NULL Return 0 ELSE Return totalNodes(TREE->LEFT) + totalNodes(TREE->RIGHT) + 1 [END OF IF] Step 2: END</div></div></div>			5M								
3.	<div>Describe the properties of a good hash function. Explain Folding Method of hashing with an example.</div> <div><div>3 Properties: 3 x 1 = 3 M</div><div>1: Low cost 2: Determinism 3: Uniformity (Brief Description required)</div><div>Folding Method: Description with example. 2M</div><div>Step 1: Divide the key value into a number of parts.</div><div>Step 2: Add the individual parts. Ignore the last carry, if any.</div><div>E.g.: Size of hash table m = 100 , index: 00 - 99</div><table><tr><td>Key</td><td>567801</td></tr><tr><td>Parts</td><td>56, 78,01</td></tr><tr><td>Sum</td><td>135</td></tr><tr><td>Hash Value</td><td>35 (ignore the last carry)</td></tr></table></div>			Key	567801	Parts	56, 78,01	Sum	135	Hash Value	35 (ignore the last carry)	5M
Key	567801											
Parts	56, 78,01											
Sum	135											
Hash Value	35 (ignore the last carry)											
4.	<div>Explain working of heap sort algorithm and the data structure used for heap sort. Demonstrate heap sort in detailed steps to sort 5,3,6,1,7,4,8 in ascending order.</div> <div>Explanation of Heap sort : 5M</div> <div>Data structure Description:</div> <div><div>- Heap sort uses a tree based data structure called Heap.</div><div>- Heaps are made of Complete Binary Trees.</div></div>			10M								

- Two types of heap
 1. **Max Heap** where, key value at every node \geq key value of all its descendants. Max heap is used for ascending order sorting.
 2. **Min Heap** where, key value at every node \leq key value of all its descendants. Min heap is used for descending order sorting.

Heap Sort Description:

- Array representation for binary tree is used for heap sort.
- Given array is considered the array representation of complete binary tree.

1. Creation of Max-heap

Convert this tree into max-heap by applying heapify algorithm at every non-leaf node in a bottom-up right-to-left manner.

2. Sorting by swap and heapify

2.1 In max-heap largest value is in root, swap it with last node of the heap and exclude the swapped largest value from further processing. (Effectively moving larger values towards end of the array)

2.2 Apply Heapify at new root to make it again a max-heap for the elements remaining elements.

- Repeat step 2 until there is only one element left in the heap.

Demonstration of algorithm: **5M**

Max heap to be created for ascending order sort using array representation of tree.

