**Dept. of CSE, Dhaka University**
**CSE-2101 Data Structure**
**Incourse-1 Time: 1 Hour**

**Answer the following questions:**

| | | | |
|---|---|---|---|
| **1.** | | For each of these statements about sorting algorithms, state whether it is true or false. | **3** |
| | **a.** | Insertion sort has optimal best-case running time. | |
| | **b.** | Selection sort compares each pair of keys at most once. | |
| | **c.** | We can sort an array by date by first sorting it by day, then by month, then by year, but only if we use a stable sorting algorithm. | |
| | | | |
| **2.** | | Consider the numbers in an array: 5, 6, 8, 3, 9, 4, 7. Suppose you have decided to use quicksort to sort them (considering the $1^{st}$ element as pivot). Show the result of the first call to partition() by giving content of the array after each exchange (swap). Please show each step in a new line and write only the two elements that were exchanged. | **3** |
| | | | |
| **3.** | | Construct a BST using the values 20, 10, 26, 33, 8, 14, 11, 12 and 9. Start with an empty tree and insert the elements one by one according to the given order. Show each insertion step. Show the tree after you delete 10. | **4+2** |
| | | | |
| **4.** | | Consider the following three algorithms:<br>_ **Algorithm 1** solves problems of size *N* by recursively dividing them into 2 sub-problems of size *N*/2 and combining the results in time *c* (where *c* is some constant).<br>_ **Algorithm 2** solves problems of size *N* by solving one sub-problem of size *N*/2 and performing some processing taking some constant time *c*.<br>_ **Algorithm 3** solves problems of size *N* by solving two sub-problems of size *N*/2 and performing a *linear* amount (i.e., *cN* where *c* is some constant) of extra work. | |
| | **a.** | For each algorithm, write down a *recurrence relation* showing how *T*(*N*), the running time on an instance of size *N*, depends on the running time of a smaller instance. | **3** |
| | **b.** | For each of the algorithm, state whether T(N) is approximately equal to i. *c* ii. *c logN* iii. *cN* iv. *cNlogN* or v. $cN^2$ | **3** |
| | **c.** | For each of the following algorithms, pick which of the above classes of algorithms (1, 2, or 3) applies to that algorithm: i. Mergesort ii. Binary search in a sorted array iii. Quicksort (if partitioning always divides the array in half) | **3** |
| | | | |
| **5.** | | Show how to implement a queue using two stacks. | **5** |

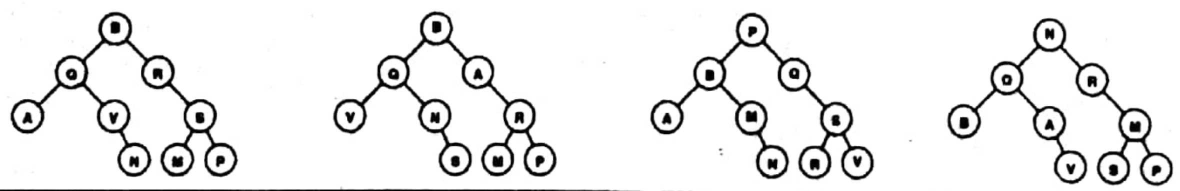
**6.** The column on the leftmost is an array of strings to be sorted. The column on the rightmost is in sorted order. The other columns (numbered 1, 2, 3 and 4) are the contents of the array at some intermediate step during one of the sorting algorithms: selection sort, insertion sort, quick sort and merge sort. Write the name of each algorithm with the corresponding column number. Use each number exactly once. **8**

| original | 1 | 2 | 3 | 4 | sorted |
|---|---|---|---|---|---|
| mars | dart | barn | barn | bark | bark |
| part | lard | care | care | barn | barn |
| care | care | fare | dart | card | card |
| gary | gary | gary | fare | care | care |
| barn | barn | harp | gary | dart | dart |
| park | card | jars | harp | earn | earn |
| rare | farm | mars | jars | fare | fare |
| fare | fare | park | mars | farm | farm |
| warm | harm | part | park | gary | gary |
| tarp | earn | rare | part | harm | harm |
| jars | jars | tarp | rare | harp | harp |
| harp | harp | warm | tarp | jars | jars |
| vary | bark | bark | vary | vary | lard |
| dart | mars | dart | warm | part | mars |
| bark | vary | earn | bark | mars | nary |
| yard | yard | harm | yard | yard | oars |
| earn | tarp | vary | earn | park | park |
| harm | warm | yard | harm | tarp | part |
| farm | rare | card | farm | rare | rare |
| tart | tart | farm | tart | tart | tarp |
| card | park | lard | card | warm | tart |
| lard | part | nary | lard | lard | vary |
| oars | oars | oars | oars | oars | warm |
| nary | nary | tart | nary | nary | yard |

**7. a.** Which of the following tree(s) will produce both the following traversals? **3**

In-order: AQVNBRMSP

Pre-order: BQAVNRSMP



**b.** If you know that a tree is a BST, which of the following is or is not always sufficient to reconstruct it? For each one, write **Yes** if it is enough to reconstruct the tree, or **No** if it is not. **3**

i. Pre-order traversal

ii. In-order traversal

iii. Post-order traversal

**Dept. of CSE, Dhaka University**
**CSE-2101 Data Structure**
**Incourse-2 Time: 1 Hour**

**Answer any 3 (three) questions**

| | | |
|---|---|---|
| **1.** | Insert the following numbers in an empty AVL tree in the given order<br>3, 2, 1, 4, 5, 6, 7, 16, 15, 14.<br>Show each insertion step and the required rotations separately. | **10** |
| **2.** | Construct a **5-ary** B tree with the following numbers. Insert them into an empty B-tree in the given sequence<br>2, 3, 44, 45, 6, 7, 55, 56, 17, 21, 68, 70, 9, 85, 18, 60, 32.<br>Show the steps clearly. | **10** |
| **3.** | Generate **Huffman code** for each character in the string "**business before pleasure**". For characters with same frequency, insert them into the queue in lexicographical order (for the first time). Write down the Huffman codes for each character (you must generate the Huffman tree first). Show each step of the tree construction. | **10** |
| **4.** | Write psudocodes for quick-find and quick-union algorithms. Discuss on the limitations of each of them.<br>Modify your psudocode for quick-union to overcome the problem of tall trees. | **7+3** |