Data Structures Concepts

1. What is a data structure and what is the need for a data structure?
2. Differentiate between linear and non-linear data structures.
3. What do you mean by abstract data types in Data Structure?
4. How is a linked list different from an array in terms of memory allocation and access time?

Arrays

1. Explain a two dimensional array and method to calculate the address of a two dimensional array. Also, write the algorithms for insertion and deletion operations at a specific position in arrays.
2. A two-dimensional array defined as A[4....7, -1....3] requires 2 bytes of storage space for each element. If the array is stored in row-major from having base address 100, then compute the address of A[6][2].
3. Write an algorithm to insert an element at any valid position in the list using the provided array representation where each element of the list is mapped to an array position using the formula: location(i) = (6-Fi) % 10.
4. Write a program to determine the count of even numbers and odd numbers in a 2D array. Declare the 2D array dynamically.

Linked Lists

1. Write an algorithm named LINK_LIST to create a linked list where each node can store a group of four digits from a given number. For instance, if the number is 23156782913287, the linked list should represent the number as follows: 3287 -> 8291 -> 1567 -> 23.
2. Assume that the node of a single linked list is in the usual data-link(next) form with the data of type int. Write C pseudocode/function to make a linked list with the following order of data: 30, 25, 72, 58, and 41. Suppose that the head is a pointer of the type node. After the initially linked list is created, the head should point to the first node of the list. Declare additional variables as you need them. After the above created a linked list, write a pseudocode/function to solve parts insert 10 before 30, insert 43 before 72, insert 8 at the end of the list, and insert 12 after 58. Lastly, what is the output of your code used to print the final list?
3. Discuss various types of linked-lists with their time-requirements for operations such as insertion at the end, searching and deleting a specific element.
4. Write a program to check if a doubly linked list of characters is a palindrome or not. A palindrome is a text which is similar if read from left-to-right and from right-to-left. For example: ABCBA, ABBA, etc.

Stacks and Queues

1. What is a stack? Write algorithms/pseudocode for the basic operations of stack.
2. What is a queue? Write algorithms/pseudocode for the basic operations of queue.
3. What is the overflow and underflow condition of a linear QUEUE implemented using an array?
4. What is the overflow and underflow condition of a circular QUEUE implemented using an array?
5. Suppose QUEUE is a circular array allocated with six memory cells: FRONT = 2, REAR = 4. Illustrate the all-intermediate contents of the circular QUEUE. QUEUE: ___, A, C, D, ___, ___. We use '___' to denote an empty memory cell for notational convenience. Describe and illustrate the queue as the following operations take place: F is added to the queue, two letters are deleted, K, L, and M are added to the queue, two letters are deleted, R is added to the queue, two

letters are deleted, S is added to the queue, two letters are deleted, one letter is deleted, and one letter is deleted.

6. What are overflow and underflow conditions in the array representation of the stack?
7. Suppose five items A, B, C, D, and E, are pushed in a stack, one after the other, starting from A. The stack is popped four times, and each element is inserted into a queue. Then, two elements are deleted from the queue and pushed back on the stack. Now, one item is popped from the stack. Estimate the popped item. Also, show all contents of the stack and queue as an intermediate step.
8. Explain the basic operations of a stack using a linked list with algorithms or pseudocode.
9. Consider a queue Q that is implemented using two stacks S1 and S2. Write an algorithm for insert(Q, item) using push and pop operations only, where a call to insert(Q, item) inserts an element at the rear of the queue.
10. Consider a queue Q that is implemented using two stacks S1 and S2. Write an algorithm for delete(Q, X) using push and pop operations only, where a call to delete(Q, X) deletes the front element and assigns it to a variable X.
11. Trace the queue algorithms to demonstrate their correctness.
12. How can a postfix expression be computed using a stack machine? Illustrate the steps of the algorithm on the output of Part (b) of Problem 2 with A=B=C=2 and D=E=F=3.
13. A game will be organized for the children attending Rahul's birthday party. Ten gift packets will be put on one another in the form of a stack on a table. A number is printed on all the gifts: {11, 7, 9, 45, 35, 46, 89, 23, 65, 90}. The gift with the 11 number is the top gift, and the number 90 is the bottom gift of the stack. A child will come near the table and will arrange the gifts in decreasing order of numbers. Write a C-based recursive code/pseudocode to help the child sort all the gifts in decreasing order using stack data structure only.
14. Explain what a circular queue is.

Expression Conversion

1. Convert the given infix expression K + L - M * N + (O^P) * W/U/V * T + Q into an equivalent prefix expression using stacks. Show the contents of the stack in each intermediate step.
2. Convert the infix expression (A + B) * (C - D) / E ^ F - G to postfix notation and evaluate the postfix expression using stacks. Show all intermediate steps in the stack given A = 2, B = 3, C = 10, D = 8, E = 2, F = 3, G = 5.
3. Consider the arithmetic expression P, written in postfix notation: P: 12, 7, 3, -, /, 2, 1, 5, +, *, +. Evaluate the expression 'P' using a stack. Visualize all the intermediate steps in the stack. Translate the expression 'P' into its equivalent infix expression.
4. Convert the infix expression A+B+C*D*ErF into its prefix and postfix equivalent forms.
5. Convert the infix expression At B*C+D-E/F/(H-G) into its prefix and postfix equivalent forms.
6. Write an algorithm to convert an infix expression into a postfix expression and demonstrate its working on A+B+C*D*EtF.
7. Explain how many PUSH and POP operations will be needed to evaluate the expression (A*B)+(C*D/E) by reverse polish notation in a stack machine.
8. Write a procedure/algorithm to convert an infix expression into a postfix expression using stack and demonstrate its working on S-FT+U*V*WI.X.
9. Convert the following infix expression to the prefix expression: (A$B + C)#(K + L — M * N + (OAP) * W IU#V * TSQ — X#Y +. Use the given precedence table. Show contents of the stack at each intermediate step.

Recursion

1. Write a C++ program to compute the nth term of the harmonic series using recursion. The harmonic series is defined as: The program should prompt the user to enter the value of n. After receiving the input, it should compute and print the nth term of the harmonic series.
2. Write a generic recursive algorithm to solve the Tower of Hanoi problem.
3. Find the sequence of moves required to complete the Tower of Hanoi task for n=4 by drawing the tree of calls/recursion tree using a recursive algorithm.
4. Comment on the running time of the Tower of Hanoi algorithm.
5. Write a recursive algorithm for Mergesort and use the following sequence of keys to demonstrate its working: 75, 80, 85, 90, 95, 70, 65, 60, 55

Algorithms and Analysis

1. What is an asymptotic analysis of an algorithm? Explain with the help of a suitable graph and write the equation for all asymptotic notations.
2. Compute the complexity of the following pseudo-codes, giving proper justifications. for(i=1; i<n; i=i*2) { for(j=i; j<n; j++) printf("x"); }
3. Compute the complexity of the following pseudo-codes, giving proper justifications. int a = 0, b = 0; for (i = 0; i < N; i++) { a = a + rand(); } for (j = 0; j < M; j++) { b = b + rand(); }
4. Compute the complexity of the following pseudo-codes, giving proper justifications. int a = 0, i = N; while (i > 0) { a += i; i /= 2; }
5. Describe various asymptotic notations by providing their definitions and examples. Sort the following time complexities in ascending order: O(n*log n), O(n^2), O(n!), O(log n), O(1), O(n^3), O(n^n), O(2^n).
6. Solve the recurrence relation T(n) = 2T(n/2) + n using the substitution method.
7. Solve the recurrence relation T(n) = 2T(n/2) + O(log n).
8. Explain what asymptotic notations are.
9. What does the term asymptotic complexity mean for any algorithm?

Searching Algorithms

1. Write an algorithm for Binary Search and discuss its limitations over linear search. Let DATA be the following sorted given elements in the array as DATA: 12, 21, 29, 31, 34, 41, 44, 55, 60, 65, 66, 77, 80, 88, 99. Apply the Binary Search algorithm to DATA to search the ITEM=85. Show all intermediate steps.
2. Differentiate between linear search and binary search in terms of their time complexity and applicability to sorted and unsorted lists.
3. Write the algorithm for Binary search and discuss its limitations and advantages over linear search.

Sorting Algorithms

1. Write the recurrence relation for the worst case of Merge Sort. Give the algorithm for the Merge Operation used in Merge Sort for sorting in a non-decreasing order. Is Merge Sort In-Place and is it stable?
2. Give the algorithm/pseudocode for the partition function in QuickSort that works well for sorted data and data with duplicate values. You may take the example [1, 1, 1, 1, 1] and show your working. Is Quick Sort an in-place sorting algorithm?
3. Show the working of an algorithm that sorts the array [2,1,0,0,1,2,0,0,1,2,1] in non-decreasing order in a single pass. What is the name of the algorithm? Is it in-place?

4. Write a C++ program to sort the provided list of numbers using the Bubble Sort algorithm. Provide a step-by-step visualization of each iteration of the sorting process. Input: List of numbers to sort: {5, 3, 8, 2, 1} Output: Sorted list: {1, 2, 3, 5, 8}. Discuss the time complexity of the Bubble Sort algorithm in terms of its worst-case, best-case, and average-case scenarios. Additionally, comment on its space complexity and any potential drawbacks associated with its performance. Discuss how Quick Sort's time and space complexity compare to Bubble Sort and highlight any advantages or disadvantages of using Quick Sort in this context.
5. Write an algorithm for merge sort and sort the given values using Merge Sort: 67, 72, 77, 80, 85, 60, 58, 50, 45. Also, solve the running time equation of merge sort.
6. Sort the following list using Insertion sort. Show each pass of the sort and discuss its run time complexity: 78, 23, 56, 33, 59, 99, 12, 10, 90, 75.
7. Write an algorithm for insertion sort and sort the given values using insertion Sort: 34, 17, 70, 28, 21, 34, 52, 41, 11. Also, solve the running time equation of merge sort.
8. Explain the quick sort algorithm.

Trees

1. What is a binary search tree?
2. Draw the binary search tree after inserting the following numbers in order into an initially empty binary search tree: 6, 5, 1, 8, 3, 4, 0, 9, 6, 2.
3. What are the pre-order, in-order, and post-order traversal sequences of the binary search tree?
4. Given the preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42, what is the postorder traversal sequence of the same tree?
5. Given the preorder traversal sequence of a binary search tree is 15, 10, 12, 11, 20, 18, 16, 19, what is the postorder traversal sequence of the same tree?
6. Suppose the numbers 7 , 5 , 1 , 8 , 3 , 6 , 0 , 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. Draw the binary search tree and what are the pre-order, in-order and postorder traversal sequence of the resultant tree?
7. Suppose the numbers 46 , 45 , 41 , 48 , 43 , 44 , 40 , 49, 56, 42 are inserted in that order into an initially empty BST. The BST uses the usual ordering on natural numbers. Draw the BST starting with an empty tree. Redraw the BST to show the effect of deletion of node with value 41. Finally, traverse the resultant BST using pre-order, in-order, and post-order traversal methods.

Heaps and Priority Queues

1. Given the array [1, 2, 3, 4, 5, 6, 7, 8], convert it into a MAX HEAP using the Sift-Down method. Show the final array representation. What is the time complexity of this process for an input size n? Justify your answer. Starting from the max heap obtained, perform the deletion of the maximum element. Show all intermediate steps.
2. Explain the process of insertion into a max heap with pseudocode.
3. Insert a new node with value 57 into the following max heap: 45, 30, 25, 15, 11, 8, 16.
4. Explain the use of Heaps as priority queues.

Graphs

1. What is a minimum spanning tree? Explain Kruskal's algorithm.
2. Explain how graphs are represented in memory.
3. Write the pseudocode for breadth first traversal in a graph.
4. Explain breadth first traversal in a graph with an example.

5. What is a minimum spanning tree? Solve the given graph using Prim's algorithm.
6. Differentiate between breadth first traversal and depth first traversal with a suitable example.
7. Write an algorithm/pseudocodes for depth first search traversal (DFST) of a graph. Apply the DFST algorithm on node 1 of the graph shown in Figure 2.
8. Explain how Prim's algorithm for finding a minimum spanning tree is different from Kruskal's algorithm. Explain the steps of Kruskal's algorithm for computing an MST for the graph shown in Figure 1.

Hashing

1. Explain hashing and types of collision resolution techniques in hashing with suitable examples.
2. Consider a hash table with 9 slots. The primary hash functions are h(k) = k mod 9. We insert the following keys in the order 15, 28, 17, 5, 10, 33, 12, 19, 20. Draw the contents of the hash table when the collisions are resolved by Open hashing, Open addressing with Quadratic Probing, and Closed hashing with Double hashing. Use secondary hash function h2(k) =7-(k mod 7)

String Matching

1. Compute the LPS array for the pattern given in Figure 3.
2. Illustrate the working of the KMP algorithm by applying it on the given instance of the string matching problem.

Memory Management

1. In some language that supports negative indexes, a C array declaration Data[9][6][6] can be written as Data[-1:7, -5:0, 7:12]. Let the base address of the array be 500 and each element occupies 4 bytes of memory. Compute the address of Data[3][-2][10] if the elements are stored in a row major order?

Database Management Systems

1. What is a database management system? What are the problems of the Hierarchical Model? Explain with examples.
2. Discuss the importance of data independence used in DBMS as opposed to the File System. Also explain different types of data independence with suitable examples.
3. Draw the three-level architecture of RDBMS based on a table with Roll_Number, Name, Mobile Number, and Email, where User 1 wants to fetch only Roll_Number and User 2 wants to fetch Roll_Number and Name of all the students. Also, discuss any two points that show the comparison among different Data models.
4. Differentiate between Left Outer Join and Right Outer Join.
5. Differentiate between Trivial and Non-Trivial Dependency.
6. Differentiate between Generalization and Specialization.
7. Explain the differences and similarities between the third normal form and Boyce Coded Normal form with examples.
8. What do you mean by Concurrency? Why is concurrency control required? Discuss with suitable examples.
9. Discuss the different types of anomalies with suitable examples.
10. Convert an ER diagram into database tables. Explicitly mention the primary and foreign keys for each of the database tables.
11. Differentiate between DROP and DELETE.

12. Differentiate between ON DELETE SET NULL and ON DELETE CASCADE.
13. Differentiate between Entity integrity constraint and Referential Integrity constraint.

Normalization

1. Consider the following functional dependencies in a database: Data_of_Birth -> Age, Age -> Eligibility, Name -> Roll_number, Roll_number -> Name, Course_number -> Course_name, Course_number -> Instructor, (Roll_number, Course_number) -> Grade. In which normal form is the relation (Roll_number, Name, Data_of_Birth)? Explain your answer in detail by considering all the rejected and accepted FDs.

Concurrency Control

1. Determine which of the following schedules are conflict serializable and explain why: S1: r1(X); r1(Y); r2(X); r2(Y); w2(Y); w1(X) and S2: r1(X); r2(X); r2(Y); w2(Y); r1(Y); w1(X).