Question Paper Analysis

1. Perform the following operations using stacks. Show contents of the stack at each intermediate step. (a) Convert the given infix expression into an equivalent postfix expression. A B C * (D + E / F G) H (b) Compute the value of the postfix expression obtained in

Topic: STACK Difficulty: 6

2. (a) for A = 45, B = F = 2, C = 5, D = 8, E = 6, G = 4, and H = 3. (2) (2)

Topic: BACKTRACKING Difficulty: 9

3. Write a complete algorithm/pseudo-code to implement any one of the following: Quicksort sorting algorithm OR Mergesort sorting algorithm (3)

Topic: SORTING Difficulty: 3

4. (a) Solve the following recurrence relation. (b) Find the recurrence relation and solve it for the function given in Fig. 1. (1) (2) 1. int power(int x, int n) 2. { if (n==0) 3. return 1; 4. else if (n==1) 5. return x; 6. else if ((n%2)==0) 7. return power(x, n/2)*power(x, n/2); 8. else 9. return power(x, n/2)*power(x, n/2); 10. } 1. for (int k = 1; k <= 7; k++) 2. Q.enqueue(k); 3. for (int k = 1; k <= 4; k++) 4. { 5. Q.enqueue(Q.dequeue()); 6. Q.dequeue(); 7. } Fig. 1 Fig. 2

Topic: COMPLEXITY Difficulty: 4

5. (a) Let f(n) = 7n + 8 and g(n) = n. Is f(n) = O(g(n))? If yes, then determine the values of n0 and c showing all intermediate steps. If no, then justify your answer with appropriate explanation. (b) An algorithm ALGO consists of two tuneable sub-algorithms ALGOA and ALGOB, which have to be executed serially. Given any function f(n), one can tune ALGOA and ALGOB such that one run of ALGOA takes time O(f(n)) and ALGOB takes time O(n/f(n)). For the given scenario, determine the smallest growing function f(n) which minimizes

the overall runtime of ALGO. (1) (2)

Topic: COMPLEXITY Difficulty: 4

6. Let Q be a circular array-based queue capable of holding 7 numbers. Execute the code snippet given in Fig. 2. After each execution of the for loop in lines 3 to 7, give the values of front pointer, rear pointer, and valid contents of Q, i.e. elements in between the front and the rear pointers. (2)

Topic: CIRCULAR ARRAY Difficulty: 6

7. Let S be an empty stack and Q be a queue having n numbers. isEmpty(Q) or isEmpty(S) returns true if Q or S is empty, else returns false. top(S) returns the number at the top of S without removing it from S. Similarly, front(Q) returns the number at the front of the queue Q without removing it from Q. Determine the best- as well as the worst-case running time of an algorithm shown in Fig. 3. Justify your answers giving suitable examples. [Hint: Use n <= 4]. (2) 1. while (!isEmpty(Q)) 2. { if (isEmpty(S) || top(S) >= front(Q)) 3. { S = push(S,front(Q)); 4. Q = dequeue(Q); 5. } 6. else 7. { Q = enqueue(Q,top(S)); 8. S = pop(S); 9. } 10. } 1. /* Integer n is the number of elements in an array A[0..n1]. */ 2. void module(int *A, int n, int k) 3. { int temp, i, j; 4. for (j = 0; j < k; j++) 5. { temp = A[n-1]; 6. for (i = n - 1; i > 0; i--) 7. A[i] = A[i - 1]; 8. A[i] = temp; 9. } 10. } Fig. 3 Fig. 4

Topic: STACK Difficulty: 5

8. Answer the following questions with respect to the function given in Fig. 4. (a) What is the purpose of designing it? [Hint: Use n <= 5, 1 <= k <= n] (b) What is its complexity? (c) Is answer to

Topic: COMPLEXITY Difficulty: 5

9. (b) dependent on the value of k? If yes, then for k > n suggest a single line modification in the given function to maintain the identified time complexity as in

Topic: COMPLEXITY Difficulty: 2

10. (b). If no, then give suitable justification with examples for the identified independency. (2)

Topic: HASHING Difficulty: 9

11. Given a singly linked list (LL1) having 2*n nodes (n (a) Write an algorithm/pseudo-code to create two linked lists (LL2 and LL3) each having n 1 nodes. LL2 and LL3 are respectively formed by adding values of consecutive odd-positioned and even-positioned nodes in LL1. Note: Position of first node in LL1 is one. Example: n = 3, LL1: 1 2 3 4 5 6 LL2: 4 8

LL3: 6 10 (b) Write an algorithm/pseudo-code to combine LL1 with LL2 and LL3 (formed in

Topic: LINKED LIST Difficulty: 9

12. (a)). Nodes of LL2 and LL3 are to be placed at alternative positions in first-half and last-half of LL1. Create a new node MID that contains sum of first and last node values of LL1 and place it in the middle of the updated LL1 as shown in Fig. 5. Note: Creation of new node is not allowed, only reposition the existing nodes. Example: In continuation with example of

Topic: BST Difficulty: 9

13. (a) MID: 7 Updated LL1: 1 4 2 8 3 7 4 6 5 10 6 LL2: NIL and LL3: NIL (6) Fig. 5

----------ALL THE BEST----------

Topic: HASHING Difficulty: 9