



Daffodil International University
Faculty of Science & Information Technology
Department of Computer Science & Engineering
Midterm Examination, Fall 2026
Course Code: CSE213 , Course Title: Algorithms
Level: 2 Term:1 Batch: 70

Time: 01:30 Hrs

Marks: 25

Answer ALL the Questions

[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1.	a)	Explain the relationship between time complexity and space complexity. Discuss how memoization and caching can reduce execution time at the cost of additional memory, using suitable examples.	02✓	CO1
	b)	Differentiate between Big-O, Big-Ω, and Big-Θ notations in terms of their asymptotic bounds. Determine the asymptotic complexity of the following function using all three notations: $f(n) = 4n^3 + 7n^2 + 6n + 12$	03✓	
2.	a)	Analyze the time and space complexity of Binary Search. Compare its average case performance with Linear Search.	02✓	CO2
	b)	Analyze the time complexity of the following code snippet and express it using asymptotic notation: <div style="display: flex; justify-content: space-between;"> <pre>int main() { int n; scanf("%d", &n); int i = 1; while (i <= n) { int j = 1; while (j <= n) { printf("*"); j++; } i = i * 2; } return 0; }</pre> <pre>#include <stdio.h> int main() { int n; scanf("%d", &n); for (int i = 1; i < n; i *= 2) { for (int j = 1; j < i; j *= 2) { for (int k = 1; k < j; k++) { printf("Hello\n"); } } } return 0; }</pre> </div>	1.5*2 = 03✓	
3.	a)	Given coin denominations {2, 3, 5, 7}, design a Greedy algorithm to make change for an amount 118. Does the greedy strategy work here! Why?	05	CO3
	b)	A warehouse has bins, each with a capacity of 10 units. The following items must be packed into the bins: 7, 5, 6, 2, 4, 3, 8. Using the First-Fit Decreasing (FFD) algorithm: <ul style="list-style-type: none"> Show step-by-step how the items are assigned to bins. Determine the total number of bins required and calculate the unused capacity in each bin. 	05	

c)	<p>Consider the following pseudocode for the Merge Sort algorithm: ✓</p> <pre> MERGE_SORT(A, left, right) if left < right mid = (left + right) / 2 MERGE_SORT(A, left, mid) MERGE_SORT(A, mid + 1, right) MERGE(A, left, mid, right) </pre> <p>For the array : $A = [38, 12, 27, 43, 9, 31, 18, 25]$</p> <ul style="list-style-type: none"> • Trace the recursive calls and show how the array is divided into subarrays until each subarray contains one element. ✓ • Simulate the execution of the MERGE() procedure and show the array after each merging step. ✓ • Write the final sorted array after the algorithm terminates. • Determine the time complexity of the algorithm in the best, average, and worst cases. 	01 02 01 01	CO3
----	---	----------------------	-----