



United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Midterm Exam Total Marks: **30** Fall-2024

Course Code: CSE2217

Course Title: Data Structure and Algorithms II

Time: 1 hour 30 minutes

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.

There are **Four** questions. **Answer all of them.** Show all the calculations/steps, where applicable. Figures in the right-hand margin indicate full marks.

1	<p>(a) Given an array $A = \{-4, 2, 0, -10, 6, -8, 14, -11\}$, find the Second Maximum number from that array using the Divide and Conquer approach. You must draw the recursion tree and clearly show each tree node's left, right, and merge steps.</p> <p>(b) Given two numbers b and n, now design a Divide and Conquer algorithm to calculate the value of b^n. Your algorithm must run in $O(\log n)$ time.</p>	<p>[4]</p> <p>[3]</p>																					
2	<p>(a) You are given a scenario where you need to provide the least number of tickets for a person to use public transportation. The available ticket options are:</p> <ul style="list-style-type: none">• Type A: 1-day pass (costs \$1)• Type B: 3-day pass (costs \$2.5)• Type C: 5-day pass (costs \$4.25)• Type D: 6-day pass (costs \$5) <p>You have an infinite number of tickets for each type. The person needs to travel for exactly 8 days.</p> <p>Using a dynamic programming approach, determine the minimum number of passes (tickets) required for the person to cover exactly 8 days. If it's not possible to cover exactly 8 days using the available passes, state "Not Possible".</p> <p>(b) What are the key characteristics of a problem that make it suitable for a dynamic programming approach?</p> <p>(c) Give an example of a problem where a greedy algorithm may fail, but a dynamic programming solution works.</p>	<p>[5]</p> <p>[2]</p> <p>[2]</p>																					
3	<p>(a) From the table below, consider the instance of the fractional knapsack problem with maximum capacity = 13. Specify which objects should be placed into the knapsack, and the maximum possible total profit. Also, explain why your solution satisfies the <i>Optimal Substructure</i> property.</p> <table><tr><td></td><td>Object 1</td><td>Object 2</td><td>Object 3</td><td>Object 4</td><td>Object 5</td><td>Object 6</td></tr><tr><td>Weight</td><td>2</td><td>5</td><td>6</td><td>4</td><td>7</td><td>3</td></tr><tr><td>Profit</td><td>1</td><td>12</td><td>14</td><td>10</td><td>16</td><td>7</td></tr></table> <p>(b) "Activity selection problem may have multiple optimized solutions"- True or False? Justify your answer with an example.</p>		Object 1	Object 2	Object 3	Object 4	Object 5	Object 6	Weight	2	5	6	4	7	3	Profit	1	12	14	10	16	7	<p>[4+1]</p> <p>[2]</p>
	Object 1	Object 2	Object 3	Object 4	Object 5	Object 6																	
Weight	2	5	6	4	7	3																	
Profit	1	12	14	10	16	7																	

