United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Midterm Exam Total Marks: **30** Fall-2023

Course Code: CSE2217 Course Title: Data Structure and Algorithms II

Time: *1 hour 45 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

(a) Given an array A = {-2, 3, -1, 3, -4, 4}, find the **maximum-sum continuous subarray** using divide-and-conquer approach. You must show the recursion tree and clearly mention left, right and crossing sum for each tree node. **[3]**

(b) Find out a good asymptotic upper bound on the following recurrence: **[2]**
$T(n) = 3T(n/4) + O(n^2)$.
You may use **Recursion-tree** or **Master method** to solve the recurrence.

(c) Given an array of integers A = {1, 3, -5, 2, -3, -2}, find the **Maximum** and **Minimum** using divide-and-conquer. Show the necessary steps to support your answer. **[2]**

2

(a) After obtaining your BSCSE degree, you embarked on an entrepreneurial journey and established your own thriving software company. You've been consistently successful in securing projects from a variety of clients, ensuring a steady flow of profits. However, suddenly a situation arises where your decision-making and leadership skills are put to the test. **[4]**

You have just received **5 project offers** from different clients, but you have only **7 days** to complete the projects. Your project manager prepares the following estimates for each of the projects and presents them to you for your decision.

Net Profit (In Million Dollars)	200	150	100	50	300
Duration (In Days)	3	2	1	2	5

Being an adept CSE graduate, you decide to approach the problem using **dynamic programming**. Determine which of the projects can be taken to **maximize the net profit**.

Note that you cannot partially complete a project. Also, you are unable to work on two projects at the same time on a particular day.

(b) Both the algorithmic paradigms: Divide-and-Conquer and Dynamic Programming solve a problem by breaking it into smaller problem instances, and by solving them. What is the **fundamental difference** between these two paradigms? **[2]**

	(c) You have found a treasure containing an infinite supply of $23, $16, $9, and $1 Coins. To go back home you need to pay a total of $25. However, despite having an infinite supply of coins you want to pay using the **minimum** number of coins. Give **an example** where the **greedy approach** does not provide an optimal solution in this matter.	**[2]**

3	**(a)** Derive the **best-case** and the **worst-case** running time equations for the following function **specialTask** and represent using Asymptotic Notation.
	
	`1 void specialTask(int arr[], int n) {`
	`2  for(int i=0; i<n; i++) {`
	`3  for(int j=n-1; j>i; j--) {`
	`4  if(arr[j]<arr[j-1]) {`
	`5  swap(arr[j], arr[j-1]);`
	`6  }`
	`7  }`
	`8  printArray(arr, n);`
	`9  bool flag = true;`
	`10  for(int j=n-1; j>0; j--){`
	`11  if(arr[j]<arr[j-1]) {`
	`12  flag = false;`
	`13  }`
	`14  }`
	`15  if(flag == true) {`
	`16  return;`
	`17  }`
	`18  }`
	`19  return;`
	`20 }`
	
	(b) Derive the **exact-cost equation** for the running time of the following function **funCTION** and find the time complexity in Big-Oh notation.
	
	`1 int funCTION(int n, int m) {`
	`2  int sumUP = 0;`
	`3  for(int i=1; i<=n; i=i+2) {`
	`4  int r = m;`
	`5  while(r>=1) {`
	`6  sumUP = sumUP+r;`
	`7  r = r/3;`
	`8  }`
	`9  }`
	`10  return sumUP;`
	`11 }`
		[5]
		
		[3]

4	**(a)** Find an optimal solution to the **fractional knapsack** instance of n = 5, W = 7, (v1, v2, v3, v4, v5) = (50, 30, 35, 60, 25), and (w1, w2, w3, w4) = (2, 2, 1, 3, 2). **Explain** why your solution satisfies the *optimal substructure property*.	**[3+1]**

	(b) What is **activity selection** problem? Is it true that the activity selection problem has one unique optimal solution? Justify your answer.	**[1]**
	(c) You are given the arrival and the departure times of eight trains for a railway platform, and each one is in the format: [arrival time, departure time). Only one train can use the platform at a time. Suppose that you have got the following train-use requests for the next day. { [8, 12), [6, 9), [11, 14), [2, 7), [1, 7), [12, 20), [7, 12) , [13, 19) } Find the **maximum number of trains** that can use the platform without any collision by using *earliest departure time*.	**[2]**