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 **(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. **[4]**

(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. **[3]**

2 **(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: **[5]**

- 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)

You have an infinite number of tickets for each type. The person needs to travel for exactly 8 days.

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".

(b) What are the key characteristics of a problem that make it suitable for a dynamic programming approach? **[2]**

(c) Give an example of a problem where a greedy algorithm may fail, but a dynamic programming solution works. **[2]**

3 **(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 *Optimal Substrucuture* property. **[4+1]**

	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

(b) "Activity selection problem may have multiple optimized solutions"- True or False? Justify your answer with an example. **[2]**

4 **(a)** Find the best case and worst case time complexity of the following code and represent them using asymptotic notation. **[4]**

```
int func(int A[], int B[], int C[]){
    n = A.length
    m = B.length
    for(i = 0 ; i < n ; i++){
        C[i] = A[i] + 10
    }
    sum = 0
    for(j = 0; j < m ; j++){
        if(B[j] < C[j] ){
            sum += B[j] + C[j]
        }
        sum = A[j]
        if( sum < 0 ){
            return 0
        }
    }
    return sum
}
```

(b) Suppose, A problem X of size n can be divided into **3** subproblems each of size n/2, each of the problem can be solved recursively in time T(n/2) respectively. The cost of dividing the problem and combining the results of the subproblems is *O**(**n**)*. If the size of the problem is less than or equal 2, then it can be solved in constant time. **[3]**

a. Formulate the recurrence relation for time
b. Solve the recurrence relation