United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Midterm Exam Total Marks: **30** Fall 2022

Course Code: CSE 2217 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 full simulation/tabulations wherever necessary. Figures in the right-hand margin indicate full marks.

1.	**(a)** Suppose, A problem X of size n can be divided into three subproblems each of size n/4, each of the problem can be solved recursively in time T(n/4) respectively. The cost of dividing the problem and combining the results of the subproblems is O**(**nlogn). **Formulate** the recurrence relation assuming, T(1) = O(1).	**[1.5]**
	(b) Solve the following recurrence equation: T(n) = 3T(n/3)+O(1), where T(1) = O(1).	**[2.5]**
	(c) Given an array of integers **A** = {2, -3, 2, -4, 1, -3, -2}, find the ***Maximum-sum Continuous Subarray*** using divide-and-conquer. You must show the **recursion tree** and clearly mention **left, right and crossing sum** for each tree node.	**[3]**
2.	**(a)** Following items are available in a grocery shop: ➢ 10 kilogram rice grain which costs 800 taka ➢ 10 kilogram salt which costs 890 taka ➢ 8 kilogram saffron powder which costs 2000 taka and ➢ 4 kilogram sugar which costs 500 taka A group of thieves (Thief 1, Thief 2, … Thief M) have come to steal from that shop, **each with a knapsack of capacity 8 kg**. The thieves are entering in serial, *Thief 2* enters after *Thief 1* is done with stealing, *Thief 3* enters after *Thief 2* is done with stealing and so on. *Since each thief wants to maximize his/her profit*, **how many thieves** will be needed in the group to empty the grocery shop and **what are the items** that each of those thieves carry? Show details of the calculation.	**[3]**
	(b) A document to be transmitted over the internet contains the following characters with their associated frequencies as shown in the following table: There are a total of 1000 characters in the document. I. **Build** the Huffman code tree for the message and find the **codeword** for each character. II. Decode "**0110001111**" using the codewords generated in (i).	**[3+1]**
3.	**(a)** Suppose you have computed a **Fibonacci** series using **dynamic programming**. **Justify** the following statements with **an example**: I. **Overlapping Subproblems** property has been satisfied in your computation. II. Dynamic programming gives you a **more efficient** solution than an obvious recursive algorithm.	**[1.5* 2 =3]**

Character	A	B	C	D	F	T	–
Frequency	40	23	8	10	4	12	3

	(b) What is 'Optimal Substructure' property? How does Dynamic Programming differ from Divide-and-Conquer problems in terms of handling subproblems?	**[2]**
	(c) Suppose, CoffeeLand Coffee Shop charges **50 BDT** (Bangladesh Taka) for each cup of small Americano with an additional vat of **3%**. You bought 2 cups of small Americano and gave the cashier **110 taka**. The cashier has got a huge supply of the following types of coins: **1 taka**, **2 taka, and 5 taka** in the cashbox. You don't want to carry many coins, so you asked the cashier to return the change using a **minimum number of coins**. Determine **the number** and **type of coins** the cashier should return in this scenario by applying the **Dynamic Programming** Approach.	**[3]**

4.	**(a)** Derive the best-case and the worst-case running-time equations for the following function ***calculate*** and represent using Asymptotic Notation.
	`void calculate(int n, int p, int A[]){`
	`  int prod = 0;`
	`  for (int i = 1; i<=n; i++){`
	`  for (int j = 1; j <= i*i; j++){`
	`  prod *= pow(i,j);`
	`  }`
	`  }`
	`  for(int m = 2; m <= p;m++){`
	`  if(A[m] < 100 ){`
	`  break;`
	`  }`
	`  prod = prod * A[m];`
	`  }`
	`  cout<<prod<<endl;`
	`}`
		[4]

	(b) Derive the exact-cost equation for the running-time of the following function and show that the time complexity is $O(n \log n \log_5 n)$:
	`void funFunction(int n)`
	`{`
	`  int sum = 0;`
	`  for (int k = 0; k < n; k*=2){`
	`  for (int j = n/2; j <=n; j++){`
	`  for (int i = n; i >=1; i=i/5){`
	`  sum += (i+j+k);`
	`  }`
	`  }`
	`  }`
	`  cout<<sum<<endl;`
	`}`
		[4]