INSTITUTE OF DATA SCIENCES
UNIVERSITY OF ENGINEERING & TECHNOLOGY, LAHORE
Subject: Discrete Mathematics

ASSIGNMENT- CLO 2, CLO 3

Instructions: Attempt each question only in the space provided. **ANSWERS MUST BE WRITTEN BY PRINTING ON SINGLE-SIDED A4 SHEETS ONLY.** Clearly mention the main steps of your solution. No rough sheets are to be attached.

- **ADD A FRONT PAGE.**

Case Study 1: Secure Digital Access System (CLO2-10 Marks) Auniversity deploys a secure digital access system for its libraries and laboratories. Each student is assigned a numerical ID and a 4-digit PIN. To reduce storage and improve security, the system does not store IDs or PINs directly. Instead, it relies on modular arithmetic to compute check digits and transformed values for fast verification and error detection. Authentication keys must be relatively prime, verified using the Euclidean Algorithm, and advanced counting principles are used to estimate the number of possible PINs, ensuring resistance against brute-force attacks.

Questions

1. A student ID N = 58341 is verified using a check digit computed modulo 9. Determine the check digit and explain whether an error will be detected if the ID is mistakenly entered as 58314.

2. During authentication, the system checks whether two numbers are congruent modulo 1000. Verify whether $123456 \equiv 456 \pmod{1000}$ and explain why modular congruence is efficient for verification.

3. Two authentication keys are generated as 84 and 55. Use the Euclidean Algorithm to determine whether these keys are relatively prime and explain why this property is important for security.

4. The system uses a 4-digit PIN where each digit ranges from 0 to 9. Determine how many distinct PINs are possible and explain how this is an application of advanced counting principles.

5. Express the division of -23 by 6 in the form $a = dq + r$ with $0 \le r < d$, and explain why the remainder must be non-negative in modular arithmetic.

Case Study 2: Algorithm Design and Complexity Analysis (CLO3-10 Marks) A software company is developing a data processing module that must efficiently handle large input sizes. The module includes searching, sorting, and numerical computation components implemented using both iterative and recursive algorithms. To ensure scalability, the developers analyze loop-based algorithms, nested iterations, and logarithmic growth patterns. Recursive procedures such as computing the greatest common divisor, binary search, and merge sort are also evaluated using recurrence relations. This case study demonstrates how algorithm design, recursion, and time-complexity analysis guide efficient software development.

Questions:

1. A loop in the system is defined as:

```
for (i = 1; i < n; i = i * 2)
```

Determine how many times the loop executes and analyze its time complexity.

2. A data validation routine uses the following nested loop structure:

```
for (i = 0; i < n; i++)

 for (j = 0; j < i; j++)
```

Calculate the total number of executions of the inner loop and determine the overall time complexity.

3. The system computes the greatest common divisor of two integers using the recursive algorithm:

$$\gcd(a,b) = \gcd(b \bmod a, a)$$

Explain why this algorithm terminates.

4. A recursive binary search algorithm is used to locate an element in a sorted list of n elements. Formulate the recurrence relation for its running time and solve it to obtain the time complexity.

5. The sorting component of the system uses merge sort to arrange large datasets. Write the recurrence relation for merge sort and explain why its time complexity is O(nlogn)