Bangladesh Agricultural University, Mymensingh

B.Sc. in Bioinformatics Engineering
Level-2, Semester-2
Course Code & Title: CSM 2221 Bioinformatics Algorithms
Assignment
Time: During the Lab final

Instructions for Students:

1. Answer the questions based on your student ID:
   - If your ID % 2 == 0, solve all even-numbered questions (e.g., 0, 2, 4, 6, 8).
   - If your ID % 2 != 0, solve all odd-numbered questions (e.g., 1, 3, 5, 7, 9).
2. Your report must be written like a formal lab report, including the following sections for each problem:
   - Problem Statement
   - Overview / Objective
   - Code
   - Output Screenshot
   - Discussion

   Formatting requirements:
   - Font: Times New Roman
   - Font Size: 12
   - Text: Fully justified
3. Submission Instructions:
   - Compile all report content into one PDF file, renamed with your ID (e.g., 123456.pdf).
   - Submit both the PDF report and all .cpp files, renaming the .cpp files according to the problem number (e.g., 0.cpp, 1.cpp, etc.).
   - Place the PDF and all .cpp files in one folder, renamed with your ID.
   - Do not include any .o or .exe files.
   - Compress the folder into a zip file, and submit the zip file to Google Classroom.

N.B:

- *If you have any confusion or need clarification, contact me before submission.*
- *Any form of plagiarism—from websites, AI tools, or other students—is strictly prohibited and will be punished.*

---

0. A bioinformatics database maintains a sorted list of gene identifiers to allow fast retrieval during sequence analysis. Given a target gene identifier and the sorted list stored in a file, design a program using the divide and conquer–based binary search technique to determine whether the target gene exists in the database. The program should read the number of gene identifiers followed by the sorted identifiers and the target gene from the input file. If the gene is found, its index position (0-based) should be reported; otherwise, an appropriate message should be produced. The result must be written to an output file.

Input Format:
The input file contains three lines: the first line contains an integer N representing the number of gene identifiers, the second line contains N space-separated gene identifiers in sorted order, and the third line contains the target gene identifier to be searched.

Output Format:
The output file contains either the index position of the target gene if it is present in the list or a message indicating that the gene identifier is not found.

Sample Input (input.txt):

	8 BRCA1 CFTR EGFR KRAS MYC PTEN TP53 VEGFA TP53 **Sample Output (output.txt):** Gene found at index: 6
1.	A bioinformatics analysis system receives gene expression values from multiple experiments, where the data may already be partially sorted due to prior preprocessing steps. Given a list of expression values stored in an input file, design two sorting modules using Merge Sort and Quick Sort. In addition to sorting the data, each module must count the number of key comparisons performed. To analyze algorithm behavior under different data conditions, the Quick Sort implementation must always select the first element as the pivot. After execution, the program should output the sorted sequence, the comparison count for each algorithm, and a conclusion stating which algorithm performs better for the given input. All results must be written to an output file. **Input Format:** The input file contains two lines: the first line contains an integer N representing the number of gene expression values, and the second line contains N space-separated integers, which may be partially sorted. **Output Format:** The output file should contain the sorted list of expression values, the total number of comparisons made by Merge Sort, the total number of comparisons made by Quick Sort (first-element pivot), and a final comparison statement. **Sample Input (input.txt):** 8 5 10 15 20 3 8 25 30 **Sample Output (output.txt):** Sorted Values: 3 5 8 10 15 20 25 30 Merge Sort Comparisons: 17 Quick Sort Comparisons: 28
2.	A bioinformatics lab is analyzing a gene interaction network where each gene is represented as a node and interactions are directed and weighted. The network may also contain disconnected components. Given the network stored in an input file, design a program that performs both Breadth-First Search (BFS) and Depth-First Search (DFS) starting from a specified gene. The program should record the traversal order and the cumulative weight of edges traversed. For disconnected components not reachable from the starting gene, the program should perform additional traversals to explore the entire network. The output file should report the BFS and DFS traversal sequences along with the total cumulative interaction weight for each traversal, helping researchers understand connectivity and interaction strength across the network. **Input Format:**

	The input file contains multiple lines. The first line contains two integers N and M, representing the number of genes and the number of interactions. The next M lines each contain three integers: the source gene, the target gene, and the interaction weight. The last line contains the starting gene for traversal. **Output Format:** The output file should first display the BFS traversal sequence of genes in the order they are visited, followed by the total cumulative interaction weight for BFS. Next, display the DFS traversal sequence and the total cumulative interaction weight for DFS. Finally, include a concluding statement summarizing the network connectivity explored, noting any disconnected components. **Sample Input (input.txt):** 6 7 1 2 5 1 3 3 2 4 2 3 4 4 4 5 1 5 6 6 6 3 2 1 **Sample Output (output.txt):** BFS Traversal: 1 2 3 4 5 6 Total BFS Weight: 21 DFS Traversal: 1 2 4 5 6 3 Total DFS Weight: 21
3.	A bioinformatics lab models a gene interaction grid as a 2D matrix, where each cell represents a gene site, and the value in each cell indicates the interaction cost to move into that site. Movement is allowed to the four neighboring cells (up, down, left, right). Given the grid stored in an input file, design a program using Dijkstra's algorithm to find the minimum total interaction cost path from the top-left cell (0,0) to the bottom-right cell (N-1, M-1). The program should read the grid dimensions and cell costs from the input file and output the path taken and the total interaction cost. **Input Format:** The input file contains multiple lines. The first line has two integers N and M, representing the number of rows and columns of the matrix. The next N lines contain M space-separated integers representing the interaction cost of each cell in the grid. **Output Format:** The output file should first display the sequence of cell coordinates (row, column) representing the shortest path from the top-left to bottom-right. Next, display the total interaction cost of the path. Finally, include a concluding statement summarizing the path found.

	Sample Input (input.txt): 3 3 1 3 1 1 5 1 4 2 1 **Sample Output (output.txt):** Shortest Path: (0,0) -> (0,1) -> (0,2) -> (1,2) -> (2,2) Total Interaction Cost: 7
4.	A bioinformatics lab is constructing a minimal-cost gene interaction network for experimental analysis. Each interaction has a cost, and the lab wants to compare two strategies for building the network: Prim's algorithm, which grows the network starting from a specific gene, and Kruskal's algorithm, which adds edges in ascending order of cost. In addition to computing the MST for both methods, the program must count the number of edge comparisons made during execution for each algorithm. This allows researchers to evaluate not only total cost but also algorithmic efficiency in terms of comparisons. The program should read the network from an input file, compute MSTs using both algorithms, count comparisons, and output the results along with a final comparison based on the number of comparisons. **Input Format:** The input file contains multiple lines. The first line contains two integers N and M, representing the number of genes and interactions. The next M lines contain three integers: source gene, target gene, and interaction cost. The last line contains an integer representing the starting gene for Prim's algorithm. **Output Format:** The output file should display the MST edges generated by Prim's algorithm, followed by the total cost and the number of comparisons made. Next, display the MST edges from Kruskal's algorithm, with its total cost and comparison count. Finally, include a concluding statement comparing the two algorithms based on the number of comparisons, highlighting which algorithm was more efficient for the given network. **Sample Input (input.txt):** 6 9 1 2 7 1 3 9 1 6 14 2 3 10 2 4 15 3 4 11 3 6 2 4 5 6 5 6 9 1 **Sample Output (output.txt):**

	Prim's MST Edges: 1-2 7 1-3 9 3-6 2 4-5 6 5-6 9 Total Cost: 33 Prim's Comparisons: 12 Kruskal's MST Edges: 3-6 2 4-5 6 1-2 7 1-3 9 5-6 9 Total Cost: 33 Kruskal's Comparisons: 15
5.	A bioinformatics lab is comparing two DNA sequences to identify differences. Given two sequences stored in an input file, design a program using Dynamic Programming to compute the minimum number of operations (insertions, deletions, substitutions) required to convert the first sequence into the second. For a slight twist, assume that substituting a nucleotide with itself has 0 cost, and all other operations have a cost of 1. The program should also output the alignment showing the operations applied for each position. **Input Format:** The input file contains two lines. The first line contains the first sequence, and the second line contains the second sequence. **Output Format:** The output file should first display the aligned sequences, indicating insertions or deletions with -. Next, output the minimum edit distance. Finally, include a concluding statement summarizing the alignment. **Sample Input (input.txt):** GATTACA GCATGCU **Sample Output (output.txt):** Aligned Sequences: G A T T A C A G C A T - G C U Minimum Edit Distance: 5
6.	Given a set of gene samples, each with a weight (processing requirement) and a value (scientific importance), and a machine with a maximum capacity, determine which samples to include to maximize the total scientific value without exceeding the capacity. Each sample must be fully included or excluded (0/1 constraint). Use Dynamic Programming to compute the optimal selection. The program should output the chosen samples, their total value, and the remaining unused capacity. **Input Format:** The input file contains multiple lines. The first line contains an integer N, the number of gene samples. The next N lines each contain two space-separated integers representing the weight and value of each sample. The last line contains an integer representing the maximum capacity. **Output Format:**

	The output file should first list the selected samples with their weight and value. Next, display the total scientific value of the selection and the remaining capacity. Finally, include a concluding statement summarizing the selection. **Sample Input (input.txt):** 4 10 60 20 100 30 120 25 75 50 **Sample Output (output.txt):** Selected Samples: Weight: 20, Value: 100 Weight: 30, Value: 120 Total Scientific Value: 220 Remaining Capacity: 0
7.	Given a sequence of gene labels represented as strings, determine the Longest Increasing Subsequence (LIS) based on lexicographical order. Design a program using Dynamic Programming to compute both the length of the LIS and the actual subsequence. For a twist, repeated gene labels are not allowed in the subsequence. The program should read the sequence of labels from an input file and output the LIS along with its length. **Input Format:** The input file contains two lines. The first line contains an integer N, the number of gene labels. The second line contains N space-separated strings representing the gene labels. **Output Format:** The output file should first display the Longest Increasing Subsequence of labels, followed by the length of the subsequence. Finally, include a concluding statement summarizing the result. **Sample Input (input.txt):** 7 BRCA1 CFTR EGFR KRAS MYC PTEN TP53 **Sample Output (output.txt):** Longest Increasing Subsequence: BRCA1 CFTR EGFR KRAS MYC PTEN TP53 Length of LIS: 7
8.	Given a directed weighted graph representing gene interactions, where each node is a gene and each edge has a cost (e.g., experimental or interaction cost), compute the minimum-cost path from a given starting gene to a target gene using Dynamic Programming. The program should read the number of genes, the edges with their costs, the start and target genes from an input file, and output the path and total cost. This approach should handle graphs without negative cycles. **Input Format:** The input file contains multiple lines. The first line contains two integers N and M, the number of genes (nodes) and interactions (edges). The next M lines each contain three integers: source gene, target gene, and interaction cost. The last line contains two integers: the starting gene and the target gene. **Output Format:** The output file should first display the minimum-cost path as a sequence of gene IDs separated by arrows (->). Next, display the total cost of this path. Finally, include a concluding statement summarizing the shortest interaction path found using Dynamic Programming.

	Sample Input (input.txt): 5 7 1 2 2 1 3 5 2 3 1 2 4 2 3 4 3 3 5 1 4 5 2 1 5 **Sample Output (output.txt):** Minimum-Cost Path: 1 -> 2 -> 3 -> 5 Total Cost: 4
9.	Given a gene interaction network, where nodes represent genes and edges represent conflicting interactions, determine the minimum number of colors needed to color all genes such that no two connected genes share the same color. Each color can represent a distinct experimental condition, batch, or pathway, and minimizing colors helps optimize resources. Using backtracking, find both the minimum number of colors required and a valid color assignment for each gene. The program should read the network from an input file and output the minimum number of colors along with the coloring. **Input Format:** The input file contains multiple lines. The first line contains two integers N and M, representing the number of genes (nodes) and interactions (edges). The next M lines each contain two integers: source gene and target gene, representing an interaction. **Output Format:** The output file should first display the minimum number of colors required, followed by the color assigned to each gene in the format GeneID: Color. Finally, include a concluding statement summarizing the result. **Sample Input (input.txt):** 4 4 1 2 1 3 2 4 3 4 **Sample Output (output.txt):** Minimum Number of Colors: 3 Gene Coloring: 1: 1 2: 2 3: 2 4: 3