ASSIGNMENT - 1

CS F222 - Discrete Structures For Computer Science
Academic Year: [2025-2026, Semester 1]

Instructions

This assignment contains 10 problems designed to be solved using the C programming language. Please read the following instructions carefully:

- **Implementation:** All solutions must be written in C.
- **Lab Test Scope:** The upcoming lab test will consist of one randomly selected problem from the first nine questions (1-9) in this set.
- **Additional Problem:** Problem 10 is an optional, additional challenge problem. It is provided for further practice and exploration but will not be included in the lab test.

1. Alien Clans

You are a xenobiologist studying a new alien species. The aliens are grouped into distinct "clans" based on a complex "DNA compatibility" trait. For this clan theory to hold true, the compatibility relationship **must be an equivalence relation**.

You have the compatibility data in an `N x N` grid of 1s and 0s, called a **matrix**.

Write a program in **C language** that performs a check on this data to see if it's a valid equivalence relation and then find all the members in the clan of the alien ambassador.

The `N x N` matrix is a map of who is compatible with whom. Let's call it `matrix`.

- Each row and column corresponds to an alien, from `0` to `N-1`.
- The value at `matrix[i][j]` tells you if Alien `i` is compatible with Alien `j`.
- A `1` means **YES**, they are compatible., A `0` means **NO**, they are not.

For ex, if `N=3` and we have `matrix[0][2] = 1`, it means "Alien 0 is compatible with Alien 2."

Input Format:

- The first line: an integer `N`, the number of aliens (numbered 0 to N-1).
- The next `N` lines: an `N x N` matrix of 0s and 1s.
- The last line: an integer `k`, the ID of the alien ambassador.

Output Format:

- Print a 3-character string. The first character is '1' if the relation is reflexive, '0' otherwise. The second is for symmetric, and the third is for transitive. Example: `101` means reflexive, not symmetric, and transitive.
- If the relation is **equivalent**, print the ID's of all the aliens in the ambassador's clan.

Sample Test Case:

Input:

```
5
1 1 0 0 0
1 1 0 0 0
0 0 1 1 0
0 0 1 1 0
0 0 0 0 1
1
```

Output:

```
111
0 1
```

2. Starship Blueprint

You are the lead engineer at a spaceship factory. Building a starship is complex: the "Hyperdrive" can't be installed until the "Power Core" is in place. You have a list of all N components and a matrix of dependencies, where matrix[i][j] = 1 means "component i must be built before component j."

Your construction droids are waiting. They need a linear, step-by-step build order that respects all dependencies. Provide one valid assembly order.

Input Format:

- The first line: an integer N, the number of components.
- The next N lines: the N x N dependency matrix of a guaranteed Directed Acyclic Graph (DAG).

Output Format:

- A single line of N space-separated integers, representing one valid build order.

Sample Test Case:

Input:
6
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 1 0 0
0 1 0 0 0 0
1 1 0 0 0 0
1 0 1 0 0 0

Output: 4 5 2 3 0 1 (Other answers are possible)

3. Chain of Command

You are given an `N x N` matrix representing a military command structure (a valid **POSET**), where `matrix[i][j] = 1` means officer `i` has authority over `j`.

This chart is cluttered with redundant, transitive links. For example, it might show a General commands a Lieutenant directly, even when the path `General -> Captain -> Lieutenant` already establishes that authority.

Your task is to simplify this into a "Direct Command Chart" that shows only immediate superior-subordinate links. A link `u -> v` is a **direct command** if `u` has authority over `v`, and there is no intermediate officer `k` such that `u` commands `k` and `k` commands `v`.

Write a program to find and print all direct command links

Input Format:

- The first line: an integer `N`, the number of officers (numbered 0 to N-1).
- The next `N` lines: the `N x N` POSET matrix of authority.

Output Format:

- Print each direct command link `u -> v` on a new line. The list should be sorted first by `u` (ascending), and then by `v` (ascending).

Sample Test Case:

Input:
4
1 1 1 1
0 1 1 1
0 0 1 1
0 0 0 1

(This represents a total order, or a single chain of command: 0 -> 1 -> 2 -> 3)

Output:
0 -> 1
1 -> 2
2 -> 3

Explanation: The link `0 -> 2` is not a direct command because officer `1` is an intermediary (`0 -> 1` and `1 -> 2`). Similarly, `0 -> 3` is not direct, and so on.

4. Minimum Vigil

You are the Royal Engineer for a kingdom with `N` strategic locations. To defend the realm, you can build watchtowers. A tower built at location `i` can survey location `i` itself, plus a specific set of other locations it has a line of sight to. You have been given the complete surveillance map.

The King is frugal and wants the most efficient defense plan. Your mission is to determine the **absolute minimum number of watchtowers** you must build to guarantee that **every single strategic location** is being watched.

Input Format:

- The first line: `N` (number of locations, `1 <= N <= 20`).
- The next `N` lines: an `N x N` matrix. `matrix[i][j] = 1` means a tower at `i` can see `j`.

Output Format:

- A single integer: the minimum number of watchtowers required.

Sample Test Case:

Input:
4
1 1 0 0
0 1 1 0
0 0 1 1
1 0 0 1

Output: 2 (Explanation: Towers at locations 1 and 3 provide full coverage {0,1,2,3}.)

5. Essence and Concoction

An alchemical grimoire describes a POSET of magical reagents, where `matrix[i][j] = 1` means "reagent `i` is a component of `j`." For two reagents, `A` and `B`, you must find:

1. **The Core Essence (Greatest Lower Bound - GLB):** The most complex reagent that is a component of *both* `A` and `B`.
2. **The Grand Concoction (Least Upper Bound - LUB):** The simplest product that requires *both* `A` and `B`.

Input Format:

- The first line: an integer `N`.
- The next `N` lines: the `N x N` POSET matrix.
- The final line: two space-separated integers, `a` and `b`.

Output Format:

- `GLB: [GLB_id], LUB: [LUB_id]`

Sample Test Case:

Input:
6
1 1 1 1 1 1
0 1 0 1 1 1
0 0 1 0 1 1
0 0 0 1 0 1
0 0 0 0 1 1
0 0 0 0 0 1
3 4

Output: `GLB: 1, LUB: 5`

6. The Amulet's Incantation

You are given the 16-bit integer state of a magical amulet. To unlock its power, you must apply a sequence of transformations (an "incantation").

An incantation `T1, T2, T3` is a **function composition** and must be applied from right to left: `T1(T2(T3(state)))`. Calculate the final state of the amulet after applying the full sequence.

The Transformations:

- **`SWAP_BYTES`**: Swaps the upper 8 bits with the lower 8 bits.
  Example: `1010101111001101` becomes `1100110110101011`.
- **`ROTATE_L k`**: Performs a **circular** left shift by `k` positions.
  Example: `ROTATE_L 2` on `1011000000000001` becomes `1100000000000110`.
- **`FLIP_N k`**: Flips the `k`-th bit (from 0-15, where bit 0 is the rightmost bit).
  Example: `FLIP_N 3` on `...0000` flips the 3rd bit, resulting in `...1000`.
- **`XOR_MASK m`**: Performs a bitwise XOR with the mask `m`.
  Example: `0000111100001111 XOR_MASK 1111111100000000` results in `1111000000001111`.

Input Format

- The first line: an `initial_value` (an integer from 0-65535).
- The second line: `T`, the number of transformations.
- The next `T` lines: The name of each transformation, followed by an integer value if required.

Output Format

- A single integer representing the amulet's final state.

Sample Test Case:

Input:
43981
2
SWAP_BYTES
FLIP_N 0
Output: `52395`
Explanation: Composition is `SWAP_BYTES(FLIP_N(43981))`.

7. Triangle Stability

You are a reliability engineer for a massive server network. After analyzing several network failures, you've discovered a key vulnerability: "unstable" servers.

In your network, connections are redundant and stable if they form small, tightly-knit workgroups. A workgroup is a **3-server triangle**, where three servers are all mutually connected. A server is considered **stable** if it is part of at least one such triangle.

Conversely, a server is defined as **unstable** if it meets two conditions:

- It is connected to the network (i.e., its degree is not zero).
- It is **not part of any 3-server triangle**.

Given the complete network map, write a program to identify all of the unstable servers.

Input Format:

- The first line: `N` (number of servers) and `M` (number of cables).
- The next `M` lines: each contains two integers `u v`, representing a cable between server `u` and `v`.

Output Format:

- A single line containing the space-separated IDs of all unstable servers, sorted in increasing order. In case of no unstable servers, Output should be -1.

Sample Test Case:

Input:
5 5
0 1
1 2
2 0
2 3
3 4

Output: `3 4`

8. The Alliance Core

As a diplomat, you are given a map of alliances between two factions. To ensure stability, you must identify and remove "tenuous" alliances. An alliance is tenuous if either of the two members involved has no other allies (i.e., their degree is 1).

This removal is a cascading process: dissolving an alliance lowers the degree of its members, which may cause their *other* alliances to become tenuous. This process repeats in rounds until a stable "Alliance Core" remains, where no more alliances can be dissolved. Your task is to count the number of alliances in this final core.

Input Format:

- The first line: three integers, M (Faction A members, 0 to M−1), N (Faction B members, M to M+N−1), and K (the number of initial alliances).
- The next K lines: each contains two integers u v, representing an alliance between a member of Faction A (u) and Faction B (v).

Output Format:

- A single integer representing the number of stable alliances in the core.

Sample Test Case:

Input:
4 3 6
0 4
0 5
1 4
1 5
2 5
3 6

Output:
4

9. The Imperial Grid

You are an imperial architect tasked with drawing the master blueprint for a new city sector. The sector is a perfect rectangular grid of plazas, with R rows and C columns. Each plaza is connected by a road to its immediate neighbors (North, South, East, and West), but not diagonally.

The construction droids require a single, unified adjacency matrix for all R×C plazas. To do this, you must first map the 2D grid coordinates of each plaza to a unique, 1D plaza ID. You will use the standard **row-major order** for this mapping:

A plaza at `(row, col)` (0-indexed) is assigned the ID: `ID = row * C + col`.

Your task is to generate the complete adjacency matrix for the city grid based on this mapping.

Input Format:

- A single line with two space-separated integers, R and C (the number of rows and columns).

Output Format:

- Print the (R×C)×(R×C) adjacency matrix. Each number in a row should be separated by a space.

Sample Test Case:

Input:
2 3

Output:
0 1 0 1 0 0
1 0 1 0 1 0
0 1 0 0 0 1
1 0 0 0 1 0
0 1 0 1 0 1
0 0 1 0 1 0

10. Paths to Safety

You are given a directed graph with `n` nodes, labeled from `0` to `n - 1`. Your task is to identify all the **safe nodes** in this graph.

The rules are as follows:

- A node is a **terminal node** if it has no outgoing edges.
- A node is a **safe node** if *every possible path* starting from that node eventually leads to a terminal node.

This means a node is **not safe** if there is any path from it that could get stuck in a **cycle**, because a cycle is a path that never reaches a terminal node.

Hint 1: Think about the problem in reverse. Instead of trying to find nodes that lead to a cycle, start by identifying the nodes that are guaranteed to be safe from the very beginning.

Hint 2: Think of a solution using topo sort.

Input Format

- The first line: An integer `n`, the number of nodes in the graph.
- The next `n` lines: Each line `i` (from 0 to n-1) describes the neighbors of node `i`. It starts with an integer `c` (the count of neighbors for node `i`), followed by `c` space-separated integers representing the neighbors.

Output Format

- An array containing all the safe nodes of the graph, sorted in ascending order.

Input:

```
7
2 1 2
2 2 3
1 5
1 0
1 5
0
0
```

Output: `[2, 4, 5, 6]`