FAST – NUCES

Object Oriented Programming Lab Manual

Course Teacher	Ms. Hina Iqbal
Lab Instructor	Mr. Ahmad Jawad Mustasim
Lab TA	Ms. Mania Shakeel
Section	BSE-2B
Semester	Spring 2026

Department of Computer Science

FAST-NU, Lahore, Pakistan

Lab 3 – 2D Pointers and Jagged Array

Objectives

- Practicing regrow and shrink in 1D
- 2D Dynamic Array
- Jagged Arrays
- Introduction to smart Pointers

Read carefully and attempt following tasks

1. **Super Mario's Inventory Challenge**

   Story / Theme:
   Mario is on an adventure and has an inventory bag where he can carry limited items (like mushrooms, stars, fireballs, etc.).
   The bag (array) size is given at the start of the journey but can shrink or regrow as needed. Mario wants to be able to add items, remove items, see what's in his bag, or exit the bag. Mario is using unique int value for each item for example for mushrooms Mario will add 1 in array, for stars Mario will add 2 in array and so on.

   Instructions:

- Ask the user to input the initial size of Mario's inventory (array).
- Dynamically allocate memory using the new keyword.
- Create a menu-driven program with the following options:
  - Add item (add an integer to the array if space is available)
  - Remove item (remove the last item if the bag is not empty)
  - Display inventory (show all items)
  - Shrink inventory (reduce array size, preserving existing items if possible)
  - Regrow inventory (increase the array size)
  - Exit
- Use functions for:
  - Adding an item
  - Removing an item
  - Displaying the array
  - Shrinking the array
  - Regrowing the array
- Ensure no memory leaks — deallocate any previously allocated memory before resizing.

2. Mario’s Power Grid (2D Dynamic Square Matrix)

Story:
Mario has discovered a Power Grid hidden deep in the Mushroom Kingdom.
The grid is square-shaped, and each cell contains a power level (an integer).

Mario wants to:

- Create the grid with a size he chooses (e.g., 3x3, 4x4, etc.)
- Fill the grid with power levels
- Display the grid clearly

He may use this grid to plan his next power boost strategy!

Instructions:

- Ask the user (Mario) to input the size of the grid (N) — number of rows and columns (e.g., 3 means 3x3).
- Dynamically allocate a 2D array of size N x N using pointers and the new keyword.
- Ask the user to enter integer values for each cell in the grid.
- Display the grid in matrix form.
- Also Display Transpose of Grid Matrix.
- After displaying, free all allocated memory to avoid memory leaks.

3. Super Mario Diagonal Coin Quest

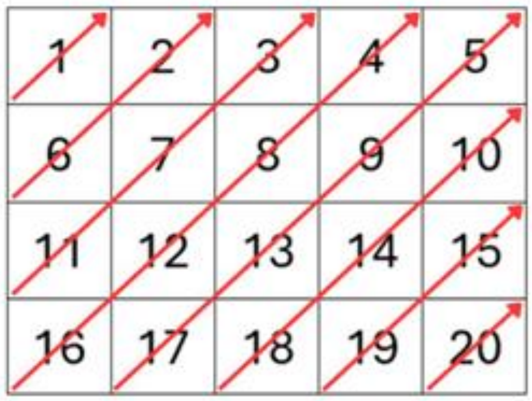
Story:
Mario is exploring a magical Mushroom Kingdom garden filled with numbered coin tiles. Each tile holds special coins, and Mario wants to collect them in a diagonal pattern to unlock a secret level.

Mario starts from the top-left corner of the garden and collects coins by moving along diagonals that go from top-left to bottom-right.
Each diagonal begins either from the first row or the first column.

Requirements

- Ask the user to enter the size of the garden (number of rows and columns).
- Create a 2D grid (garden) and take input for each tile.
- Traverse the garden diagonally:
    - Start from the top-left corner.
    - Follow diagonals from **top-left → bottom-right**.
    - Each diagonal begins from the first row or first column.
- Print all tile numbers in Mario's diagonal collection order.

Example
If the garden contains numbers from 1 to 20 arranged in 4×5 grid:



Output:
1 6 2 11 7 3 16 12 8 4 17 13 9 5 18 14 10 19 15 20

4. **Mario’s Item Chest (2D Jagged Char Array)**

Story:

Mario is now more organized on his adventure and wants to store the exact names of the items in his magical chest — like "Mushroom", "Fireball", "Star" — instead of using confusing item codes (as used in task 1).
This time, each row in the chest will store one item name, and each column will store a character (like a 2D jagged char array).
Since the length of each item name is different, we’ll use a jagged array of characters, where each row's length matches the length of the item name +1 (for the null terminator '\0').
In main(), declare an integer variable, pass its address to the function, and display the updated value using the returned pointer.

Instructions:

- Ask Mario to enter how many items he wants to store.
- For each item:
  - Ask for the name of the item.
  - Count the number of characters in the name (do not use string).
  - Allocate memory dynamically using new to store that item in a jagged 2D character array.
- After storing all the items, display the entire chest (i.e., print all item names).
- No need for adding/removing later or resizing — this is a one-time input/display task.
- Use proper memory deallocation at the end to prevent memory leaks.

Important Notes:

- You are not allowed to use string type in this task — use character arrays only.
- Remember to always include the null terminator when allocating memory for strings.
- Practice proper dynamic memory management (new / delete) to avoid memory leaks.

5. Super Mario’s Smart Pointer Adventure: Manage Mario’s Inventory Safely!

Story:

Mario’s adventure is full of powerful items — mushrooms, stars, and fireballs. To keep his inventory safe and avoid any lost or dangling items, Mario needs your help to manage his items using smart pointers! This lab will introduce you to three key smart pointers in C++: unique_ptr, shared_ptr, and weak_ptr.

By the end of this task, you’ll understand how smart pointers help automatically manage memory, avoid leaks, and safely share item ownership.

Task

Mario’s inventory holds items represented by integer codes:

- Mushroom = 1
- Star = 2
- Fireball = 3

You will help Mario manage his inventory using smart pointers as follows:

- **Unique Ownership (unique_ptr)**
  - Create a dynamic array of size N (user input) using unique_ptr<int[]> to store Mario’s inventory.
  - Write functions to add an item and remove the last item, managing the inventory within this unique ownership model.
  - Display all items currently in the inventory.
- **Shared Ownership (shared_ptr)**
  - Create individual items using shared_ptr<int>, e.g., one shared_ptr for Mushroom, one for Star.
  - Simulate sharing an item by assigning the same shared_ptr to multiple variables (e.g., Mario’s backpack and power-up list).
  - Print the use count to show how many owners an item has.
- **Weak Ownership (weak_ptr)**
  - Create a weak_ptr referencing one of the shared items to demonstrate safe access without owning it.
  - Attempt to lock the weak_ptr to access the item safely.
  - Show what happens when the original shared owners go out of scope and the item is deleted.

---------------------------------------Happy Coding ☺---------------------------------------

---------------------------------------**Submission Instructions**---------------------------------------

- Submit your tasks on Google Classroom within given deadline
- Submit solution file of each task separately for example “LAB01-TASK01.cpp”, “LAB01-TASK02.cpp” and so on
- Also submit screenshots of output of each task for example “LAB01-TASK01.png”, “LAB01-TASK02.png” and so on
- “.cpp” is the actual file where all your code is saved.
- Always submit .cpp files when asked for submission
- Example path of .cpp file is **Desktop\Your_Roll_No\LAB01-TASK01\LAB01-TASK01\LAB01-TASK01.cpp** Copy your files from this path

----