



**National University of Sciences & Technology (NUST)**  
**School of Electrical Engineering and Computer Science (SEECS)**  
**Faculty of Computing**

**CS 110: Fundamentals of Computer Programming**  
**Assignment # 2**

Assignment Date	11/11/2025	Due Date:	Sunday, 23/11/2025 (11:59 pm)	Marks: 30
CLO Attainment	[CLO-2]	Solve given real-world problem by applying appropriate programming concepts and techniques		

**Assignment Title:** Sudoku Game, Checker, and Solver

**Learning Objectives:**

By completing this assignment, you will:

- Strengthen your understanding of **two-dimensional arrays**.
- Practice **modular program design** using multiple functions.
- Learn to implement **logical constraints** for a real-world puzzle game.
- Develop problem-solving and debugging skills through **incremental feature addition**.

**Overview:**

You are tasked to develop a **Sudoku game and checker** that allows the user to:

1. **Play** Sudoku by filling in a 9×9 grid.
2. **Check** if their current board state is valid according to Sudoku rules.
3. **Request hints** to proceed correctly.
4. **Request a full solution** when desired (auto-solve).
5. Maintain and display a **score** based on the player's efficiency and number of hints used.

**Functional Requirements:**

**1. Game Setup**

- Display a **welcome message** and show an initial Sudoku board (can be partially filled or randomly generated).
- Represent the grid using a **2D array**.
- Empty cells can be represented as 0.

**2. Core Features**

- **User Interaction:**

Allow users to enter row, column, and number (1–9) to fill cells.

## FOCP Assignment 2

- **Move Validation:**  
Implement isValidMove() to ensure a number does not repeat in:
  - The same row
  - The same column
  - The same 3×3 subgrid
- **Hint System:**  
Provide a correct suggestion for one empty cell when the player requests a hint (each hint reduces score).
- **Checker:**  
On demand, check if the current grid state satisfies Sudoku rules.
- **Solver (On Demand):**  
Implement a solveSudoku() function that can auto-complete the puzzle.
- **Scoring Mechanism:**
  - +10 points for each correct placement
  - -5 for an invalid move
  - -10 for each hint
  - -20 if the full solution is requested
- **Game End:**  
When all cells are correctly filled, display a congratulatory message and final score.

### Optional Enhancements:

- Allow **difficulty levels** (Easy, Medium, Hard) with pre-filled grids.
- Allow user to enter **grid size**.
- Implement **timer-based scoring** for efficiency.
- Add **color-coded display** for better user experience.

### Concise Deliverables:

1. **Source Code (.cpp)** clean, well-commented, and modular.
2. **README.md** file with:

Section	What to Include
Title & Team Info	Project title, team members, and roles
Overview	What the game does and why it's useful
Design & Logic	List of functions, grid handling, and hint/solver approach
Execution Instructions	How to compile, run, and sample inputs/outputs
AI Tool Reflection (if any)	Mention any AI assistance used responsibly
Future Enhancements	Suggested improvements like GUI or database

3. **GitHub Repo Link**
  - Each member must contribute at least one meaningful commit.

### Evaluation Rubric (30 Marks)

Criterion	Marks	Low (1–2)	Average (3–4)	High (5)
Problem-Solving & Logic	5	Unstructured, errors	Functional but inconsistent	Clean logic, good validation
Use of 2D Arrays	5	Poor or incorrect use	Adequate handling	Efficient and modular design

## FOCP Assignment 2

Criterion	Marks	Low (1–2)	Average (3–4)	High (5)
Scoring & Hint System	5	Absent or buggy	Partially working	Smooth and well-integrated
Code Modularity	5	All in main()	Partial function use	Fully modular
Documentation	5	Minimal README	Basic README	Detailed and clear
Demo & Viva	5	Unable to explain	Partial understanding	Confident and thorough

### **Submission Criteria:**

This is a group assignment. Divide the tasks equally among each group member (all tasks have equal marks). Submit the assignment in the form of a report (pdf) on the LMS. The report should contain the following sections:

- Title page and table of contents
- Brief overview of requirements
- Screenshots of outputs/test cases
- GitHub link (Annex-I)
- Source code (Annex-II)

All group members should submit the same document in LMS (Variations are not accepted).

Name the file using the given nomenclature:

**FirstMemberName-SecondMemberName-ThirdMemberName-Assignment\_2.docx.**

Late submissions are accepted only till **Monday 24th November 2025** but with a penalty.

**Plagiarism will be marked zero.**