Summer 2025
IoT SW & HW Engineering - Field Training

جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY
كلية الحاسبات وعلوم البيانات

Assignment 1

Deadline: 17th of July, 2025 at 11:59. All team members submit on teams.

Problem 1: Conceptual Questions

Answer the following questions with references and at least one figure per each question or one diagram.

1. Define an array in C. What are its main characteristics and uses in programming?
2. Explain how memory is allocated for a one-dimensional array in C. Why is it said that arrays are stored in contiguous memory?
3. Describe how you can initialize an array in C. Give examples for full and partial initialization. What values do uninitialized elements take?
4. How do you calculate the number of elements in a statically declared array in C? Why is this technique not applicable to dynamically allocated arrays?
5. What are the consequences of accessing an array index outside its bounds? Why doesn't the C compiler always detect this error?
6. Differentiate between a one-dimensional and a two-dimensional array with examples. What are typical use cases for each?
7. How can you pass an array to a function in C? Explain what is actually passed and how this affects the original array.
8. Describe the relationship between pointers and arrays in C. How can pointer arithmetic be used to access array elements?
9. What limitations do arrays have in C, and how can these be addressed using structures like dynamic arrays, linked lists, or vectors in other languages?
10. What is the enum data type used for?

🔹 **In each coding problem, you shall submit code files, output screenshot with 3 test cases minimum.**

🔹 Problem 2: Revolving Credit Account

Write a program that:

- Accepts an account balance.
- Calculates:
    - Interest: 1.5% on first $1000, 1% on the rest.
    - Total amount due.
    - Minimum payment:
        - If ≤ $10 → pay full amount.
        - Else: greater of $10 or 10% of total.
- Includes a loop to allow repeated use until user opts out.

🔹 Problem 3: Number Manipulation

Write a program that:

- Accepts a positive integer.
- Uses the following functions:
  - isPerfectSquare() – returns true/false.
  - reverseDigits() – returns the number in reverse.
  - calculateSum() – returns the sum of digits.
- Main function:
  - Calls all three and prints results accordingly.

🔹 Problem 4: Pizza Size Comparison

Write a program to:

- Input diameter and price for two pizzas.
- Calculate cost per square inch:

$$\text{Area} = \pi \times \left(\frac{\text{diameter}}{2}\right)^2 \quad \text{Cost per sq.in.} = \frac{\text{Price}}{\text{Area}}$$

- Determine which pizza is the better buy (smaller wins in tie).

🔹 Problem 5: Longest Substring Without Repeating Characters

Task:
Given a string s, return the length of the longest substring without repeating characters.

Examples:

- Input: s = "abcabcbb" → Output: 3
  Explanation: The longest substring without repeating characters is "abc".

- Input: s = "bbbbb" → Output: 1
  Explanation: The longest substring is "b".

- Input: s = "pwwkew" → Output: 3
  Explanation: The longest substring is "wke", not "pwke" because substrings must be **contiguous** (not subsequences).

Constraints:

- $0 <= s.length <= 5 \times 10^4$

- s consists of English letters, digits, symbols, and spaces.

🔹 Problem 6: String to Integer (myAtoi)

Task:
Implement the function int myAtoi(string s) that converts the input string to a 32-bit signed integer.

Algorithm Steps:

1. **Ignore leading whitespace** characters.
2. **Check for an optional sign** ('+' or '-'). Default is positive.
3. **Convert characters to digits** until a non-digit character is encountered.
4. **Clamp the value** if it goes beyond the 32-bit signed integer range:
   $[-2^{31}, 2^{31} - 1]$.

Examples:

- Input: "42" → Output: 42
- Input: " -042" → Output: -42
- Input: "1337c0d3" → Output: 1337
- Input: "0-1" → Output: 0
- Input: "words and 987" → Output: 0

Constraints:

- 0 <= s.length <= 200
- s may include letters, digits, spaces, '+', '-', and '.'

🔹 Problem 7: Median of Two Sorted Arrays

Task:
Given two sorted arrays nums1 and nums2, return the median of the combined sorted array.
The solution must have a time complexity of O(log (m+n)).

Examples:

- Input: nums1 = [1, 3], nums2 = [2] → Output: 2.00000
- Input: nums1 = [1, 2], nums2 = [3, 4] → Output: 2.50000

Constraints:

- 0 <= nums1.length, nums2.length <= 1000
- 1 <= nums1.length + nums2.length <= 2000
- -10^6 <= nums1[i], nums2[i] <= 10^6

🔹 Problem 8: Find First and Last Position of Element in Sorted Array

Task:

Given a sorted array nums, find the starting and ending position of a given target value.
Return [-1, -1] if the target is not found.
Time complexity must be O(log n).

Examples:

- Input: nums = [5,7,7,8,8,10], target = 8 → Output: [3,4]
- Input: nums = [5,7,7,8,8,10], target = 6 → Output: [-1,-1]
- Input: nums = [], target = 0 → Output: [-1,-1]

Constraints:

- $0 <= \text{nums.length} <= 10^5$
- $-10^9 <= \text{nums[i], target} <= 10^9$
- nums is sorted in non-decreasing order.

🔹 Problem 9: Search Insert Position

Task:
Given a sorted array of **distinct integers** and a target, return its index if found.
Otherwise, return the index where it should be inserted to maintain the sorted order.
Time complexity must be O(log n).

Examples:

- Input: nums = [1,3,5,6], target = 5 → Output: 2
- Input: nums = [1,3,5,6], target = 2 → Output: 1
- Input: nums = [1,3,5,6], target = 7 → Output: 4

Constraints:

- 1 <= nums.length <= 10^4
- -10^4 <= nums[i], target <= 10^4
- nums is sorted and contains **distinct** integers.

🔹 Problem 10: Add Binary

Task:
Given two binary strings a and b, return their sum as a binary string.

Examples:

- Input: a = "11", b = "1" → Output: "100"
- Input: a = "1010", b = "1011" → Output: "10101"

Constraints:

- 1 <= a.length, b.length <= 10^4
- a and b contain only '0' and '1' characters
- No leading zeros unless the string is "0"

🔹 [Bonus] Problem 11: Word Search in a Grid

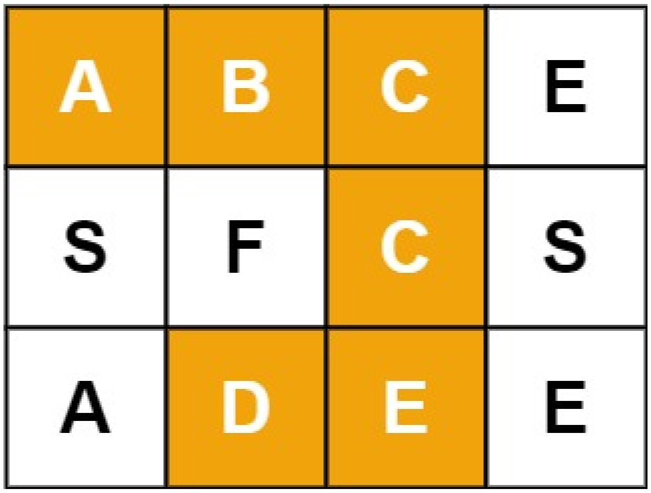
Task:
Given a 2D board of characters and a word, return true if the word exists in the grid.

Rules:

- The word must be built from **adjacent** cells (horizontal or vertical).
- A cell can only be used **once** in the path.

Examples:

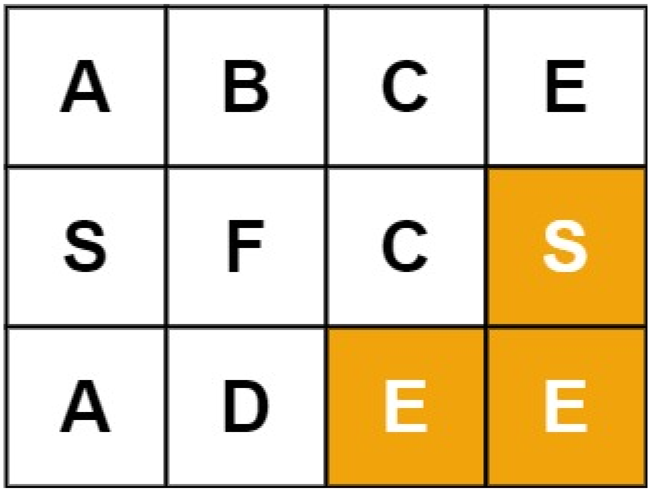
- Input:



A	B	C	E
S	F	C	S
A	D	E	E

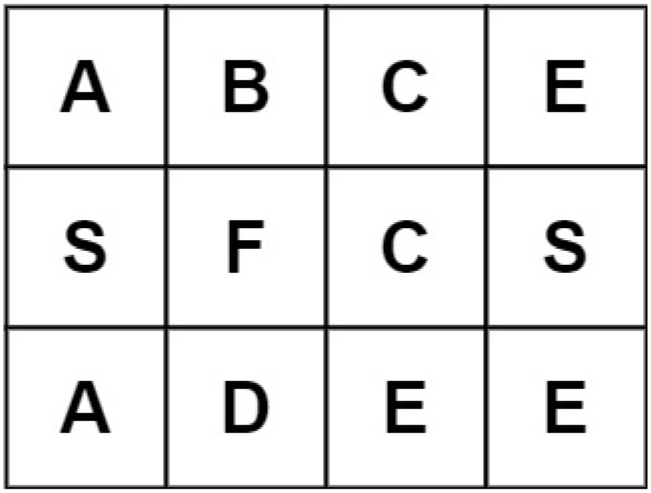
- board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"
→ Output: true

- Input:

- 

A	B	C	E
S	F	C	S
A	D	E	E

  board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"
  → Output: true

- Input:
- 

A	B	C	E
S	F	C	S
A	D	E	E

  board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"
  → Output: false

Constraints:

- 1 <= m, n <= 6 (where m = number of rows, n = number of columns)
- 1 <= word.length <= 15
- board[i][j] and word[k] consist of only English letters

- **All assignment shall be submitted as one zipped file with:**

1. Conceptual questions report.pdf
2. Coding file folder.
3. Coding file report with test cases and screens.