3rd Task in Embedded Systems

1- **Searching**
https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316079348

2- **Lowest Number**
https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316080175

3- **Sorting**
https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316080836

5- **Matrix**
https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316084844

6- **Mirror Array**
https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316085181

7- **Max Subsequence**
https://codeforces.com/group/MWSDmqGsZm/contest/219856/submission/316086305

8- **Count Words**
https://codeforces.com/group/MWSDmqGsZm/contest/219856/submission/316088420

9- **Lucky Array(Bonus)**
https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316088937

Summarization

- **Working with 1D Arrays in C:**

  A 1D array is a simple linear structure used to store multiple values of the same data type. It's useful when you need to handle a fixed-size list of items, like marks, ages.

  - **Declaration:**

  int numbers[5];

  This reserves space for 5 integers.

  - **Initialization:**

  int numbers[5] = {10, 20, 30, 40, 50};

  - **Accessing Elements:**

  printf("%d", numbers[2]);

  - **Common Operations:**

- Traversing the array using loops
- Calculating the sum and average
- Searching for an element
- Sorting

- **Handling 2D Arrays in C:**

  2D arrays are like useful tables when dealing with grid-like data such as matrices or spreadsheets.

  - **Declaration:**

  int matrix[3][3];

  - **Initialization:**

```
int matrix[2][3] = {
   {1, 2, 3},
   {4, 5, 6} };
```

- **Accessing Elements:**

  printf("%d", matrix[1][2]); // Outputs 6

- **Typical Uses:**

▪ Matrix addition, subtraction
▪ Row and column operations
▪ Transposing a matrix
▪ Multiplication of two matrices

- Understanding Strings in C

A string in C is an array of characters ending with the null character '\0' to mark the end of the string.

- **Declaration:**

  char name[6] = "Alice";

  -Note: char name[6] reserves space for 5 letters + 1 null character.

- **You can also use:**

  char name[] = {'A', 'l', 'i', 'c', 'e', '\0'};

- **Input/Output**:

  char name[50];
  scanf("%s", name);
  printf("%s", name);

- To read full lines with spaces, use fgets().

Common String Operations in C:

In C, strings are handled using character arrays, and every string ends with a null character '\0', which marks the end of the string.

- **Common String Functions:**

  -strlen(str) : Returns the length of the string (excluding '\0).
  -strcpy(dest, src) : Copies string src into dest.
  -strcat(dest, src) : Appends string src to the end of dest.
  -strcmp(str1, str2) : Compares two strings lexicographically

- **Manual Operations You Can Implement:**
  - Count characters or words in a string
  - Reverse a string
  - Convert lowercase to uppercase
  - Remove punctuation/symbols
  - Check if a string is a palindrome (reads the same forward and backward)