Task3:

What is a 1D Array?

A one-dimensional array is like a list. It stores a fixed-size sequence of elements of the same type.

Declaration and Initialization:

```
int numbers[5];  // Array of 5 integers (uninitialized)

// Initialization at declaration

int primes[5] = {2, 3, 5, 7, 11};

// Size can be omitted if initialized

float temperatures[] = {98.6, 99.2, 97.9};

// Partial initialization (remaining elements set to 0)

int scores[10] = {95, 87, 90};
```

Accessing Array Elements:

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

// Access elements using index (0-based)

printf("%d\n", numbers[0]);  // Output: 10

printf("%d\n", numbers[2]);  // Output: 30

// Modify elements

numbers[1] = 25;

numbers[3] = numbers[2] + 5;
```

Common Operations:

Iterating Through an Array:

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

// Using a for loop

for (int i = 0; i < 5; i++) {

  printf("%d ", arr[i]); }
```

```
// Using while loop
int j = 0;
while (j < 5) {
   printf("%d ", arr[j]);
   j++;
}
```

Finding Array Length:

```
int arr[] = {1, 2, 3, 4, 5};
int length = sizeof(arr) / sizeof(arr[0]);
printf("Array length: %d\n", length);  // Output: 5
```

Summing Array Elements:

```
int numbers[] = {5, 10, 15, 20};
int sum = 0;
for (int i = 0; i < 4; i++) {
   sum += numbers[i];
}
printf("Sum: %d\n", sum);  // Output: 50
```

Passing Arrays to Functions:

Arrays are passed to functions by reference (as a pointer to the first element).

```
// Function prototype
void printArray(int arr[], int size);
int main() {
   int data[] = {10, 20, 30, 40};
   printArray(data, 4);  // Pass array and its size
   return 0;
}
```

```
// Function definition
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

Common Pitfalls:

```
int arr[5];
// Wrong way to get size (in function where array decays to pointer)
void wrongSize(int arr[]) {
    int size = sizeof(arr) / sizeof(arr[0]);  // Won't work as expected
}
// Array out of bounds
arr[5] = 10;  // Undefined behavior - last valid index is 4
// Attempting to assign arrays
int a[3] = {1, 2, 3};
int b[3];
b = a;  // Compilation error
```

Important Notes:

1. Array indices start at 0, not 1.
2. C doesn't perform bounds checking - accessing out-of-bounds indices leads to undefined behavior.
3. The array name is essentially a pointer to the first element.
4. Arrays cannot be resized after declaration.

What is a 2D Array?

A 2D array is an array of arrays. It stores data in a tabular form (rows × columns).

type arrayName[rows][columns];

Example:

```
int matrix[3][4];  // 3 rows, 4 columns
```

Initializing a 2D Array:

1. Complete Initialization:

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

2. Row-by-Row Initialization

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

Accessing & Modifying Elements

Use two indices: one for row, one for column.

```
matrix[0][1] = 10;   // Set element at row 0, column 1
int x = matrix[1][2]; // Get element at row 1, column 2
```

Traversing a 2D Array

Nested for loops are used:

```
for (int i = 0; i < rows; i++) {
  for (int j = 0; j < columns; j++) {
     printf("%d ", matrix[i][j]); }
 printf("\n"); }
```

What is a String in C?

A string in C is an array of characters ending with '\0'.

char name[10] = "Alice";

This is equivalent to:

char name[10] = {'A', 'l', 'i', 'c', 'e', '\0'};

Declaring and Initializing Strings

1. Direct Initialization

char city[] = "Paris";

2. Character-by-Character

char city[6] = {'P', 'a', 'r', 'i', 's', '\0'};

Reading Strings:

1. Using scanf (reads up to whitespace)

char name[20];

scanf("%s", name);

2. Using gets() (unsafe, not recommended)

gets(name);

3. Using fgets() (safe alternative to gets)

fgets(name, sizeof(name), stdin);

Printing Strings

printf("%s", name);

Here are some useful standard functions:

Function	Description
`strlen(str)`	Returns the length
`strcpy(dest, src)`	Copies `src` to `dest`
`strcat(dest, src)`	Concatenates `src` to `dest`
`strcmp(str1, str2)`	Compares two strings
`strrev(str)`	Reverses string *(non-standard, may not be available on all compilers)*

Searching:

https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316020473

Lowest Number:

https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316022365

sorting:

https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316023754

Matrix:

https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316144081

Mirror Array:

https://codeforces.com/group/MWSDmqGsZm/contest/219774/submission/316146497

Max Subsequence:

https://codeforces.com/group/MWSDmqGsZm/contest/219856/submission/316149839

Count Words:

https://codeforces.com/group/MWSDmqGsZm/contest/219856/submission/316152529