5th Task in Embedded Systems

1. **Purpose of Double Pointers**

   Double pointers (int **ptr) are used to store the address of another pointer. They are especially useful for:

   -Dynamic memory allocation for 2D arrays: Allows creation of matrices where rows can be allocated separately

   -**Modifying pointer variables in functions**: When you need to change where a pointer points from within a function.

   -Implementing complex data structures: Used in trees, linked lists, and other advanced structures where multiple levels of indirection are needed.

2. **Relation Between Pointers, Arrays, and Strings**

   Arrays and pointers: An array name acts like a pointer to its first element.
   `int arr[] = {1, 2, 3};` is similar to `int *ptr = arr;`.
   Strings: A string is essentially a character array. `char *str = "Hello";` makes str a pointer to the first character.
   Accessing elements: You can access array or string elements using `ptr[i]` or `*(ptr + i)`.

3. **Purpose of Pointer to Function**

   A pointer to a function allows you to:

   - Pass functions as arguments.
   - Create callback mechanisms.

```
int add(int a, int b) { return a + b; }
int (*funcPtr)(int, int) = add;
printf("%d", funcPtr(2, 3));
```

4. **Accessing Arrays Using Pointers**
   - 1D Array: Access with *(arr + i) or arr[i].
   - 2D Array:
     -If declared as int arr[3][3], use *(*(arr + i) + j) to access element [i][j].
     -With dynamic allocation (int **arr), allocate memory row by row.
5. **Pointer Typecasting**

   Pointer typecasting allows converting one pointer type to another. This is useful when:

   - Interfacing with hardware or binary data.
   - Working with generic memory buffers (void * to int *). **Caution**: Incorrect casting can lead to undefined behavior.