perplexity

# this is for module 2

Here's a structured breakdown of **Module 2: Programming Questions** with code examples and theory Q&A:

## Module 2: Programming Questions Table

| Q.No | Program Description | Key Concepts Tested | Complexity Level |
|---|---|---|---|
| 1a | Count occurrences of a number in an array | Array traversal, conditional logic | Beginner |
| 1b | Display elements at odd index positions | Array indexing, loop control (`i += 2`) | Beginner |
| 2a | Display even numbers in an array | Modulus operator, array iteration | Beginner |
| 2b | Count and display odd numbers in an array | Counter variables, conditional checks | Beginner |
| 3 | Calculate average of array elements | Floating-point arithmetic, array summation | Beginner |
| 4 | Illustrate single-dimensional array initialization | Array declaration/initialization | Beginner |
| 5 | Calculate average marks of 50 students | Large array handling, floating-point division | Intermediate |
| 6 | Different ways to declare/initialize 1D arrays | Array syntax variations | Intermediate |
| 7 | Linear search implementation | Search algorithms, function design | Intermediate |
| 8 | Sort array + find largest/second largest/smallest | Bubble sort, array manipulation | Advanced |
| 9 | Bubble sort implementation | Sorting algorithms, nested loops | Advanced |
| 10 | Linear search implementation | Search algorithms, function returns | Intermediate |
| 11 | Bubble sort explanation with example | Algorithm analysis, code walkthrough | Intermediate |
| 12 | Find largest element in array | Array traversal, comparison logic | Beginner |
| 13 | Matrix row/column sum calculation | 2D array handling, nested loops | Intermediate |
| 14 | Matrix subtraction | 2D array operations, nested loops | Intermediate |
| 15 | Display diagonals of 10×10 matrix | Matrix indexing, diagonal patterns | Intermediate |
| 16 | Initialize 3×4 matrix with specific values | 2D array initialization | Beginner |

| Q.No | Program Description | Key Concepts Tested | Complexity Level |
|---|---|---|---|
| 17 | Initialize 3×4 matrix with partial values | Partial array initialization | Intermediate |
| 18 | 2D array initialization methods | Array syntax variations | Intermediate |
| 19 | Check identity matrix | Matrix validation, conditional checks | Advanced |
| 20 | Matrix transpose | 2D array manipulation | Intermediate |
| 21 | Matrix addition | 2D array operations | Intermediate |
| 22 | Matrix equality check | Deep comparison of 2D arrays | Intermediate |
| 23 | Access matrix element (2nd row, 3rd column) | Matrix indexing | Beginner |
| 24 | Read/display 3×3 matrix | Basic I/O with 2D arrays | Beginner |

## Code for Complex Programs

### 1. Bubble Sort (Q9)

```
#include <stdio.h>

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Swap elements
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

int main() {
    int arr[] = {5, 3, 8, 1, 2};
    int n = sizeof(arr)/sizeof(arr[^0]);

    bubbleSort(arr, n);

    printf("Sorted array: ");
    for(int i=0; i<n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

## 2. Identity Matrix Check (Q19)

```
#include <stdio.h>

int main() {
    int n, isIdentity = 1;
    printf("Enter matrix size: ");
    scanf("%d", &n);
    int mat[n][n];

    printf("Enter elements:\n");
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            scanf("%d", &mat[i][j]);
            if((i == j && mat[i][j] != 1) || (i != j && mat[i][j] != 0)) {
                isIdentity = 0;
            }
        }
    }

    printf(isIdentity ? "Identity Matrix\n" : "Not Identity Matrix\n");
    return 0;
}
```

## Theory Q&A

## 1. typedef vs enum

| Feature | typedef | enum |
|---|---|---|
| Purpose | Creates type aliases | Defines named integer constants |
| Syntax | `typedef int Age;` | `enum Color {RED, GREEN};` |
| Value Assignment | No values assigned | Auto-incremented from 0 |
| Use Case | Simplifying complex type names | Grouping related constants |

## 2. String Handling Functions

| Function | Purpose | Example |
|---|---|---|
| `strlen()` | Get string length | `strlen("Hello") → 5` |
| `strcpy()` | Copy string | `strcpy(dest, src)` |
| `strcat()` | Concatenate strings | `strcat(str1, str2)` |
| `strcmp()` | Compare strings lexicographically | `strcmp("A", "B") → -1` |

### 3. Matrix Initialization Methods

```
// Method 1: Full initialization
int mat[^2][^2] = {{1,2}, {3,4}};

// Method 2: Partial initialization
int mat[^3][^3] = {{1,2}, {3}, {4,5,6}};

// Method 3: Row-major initialization
int mat[^2][^3] = {1,2,3,4,5,6};
```

**Key Notes:**

- Avoid `gets()` in favor of `fgets()` for safer input handling
- Use `strlen()` for string length instead of manual counting
- Bubble sort time complexity: O(n²)
- Identity matrices require 1s on diagonal and 0s elsewhere

Let me know if you need explanations for specific programs or additional theory coverage!

⁂