perplexity

this is for module 3

read pdf (name corresponds to the module number), read through these pdfs and compile a list of all programming questions module wise and make a table of it module wise. And after list of programs, include code for complex programs and question and answers for theory questions, Avoid repetition of questions, and if one follow up question seems redundant because we have discussed same topic before in an older question, ignore that.

Here's a structured breakdown of **Module 3: Programming Questions & Theory** based on the PDF content:

Module 3: Programming Questions Table

Q.No	Program Description	Key Concepts Tested	Complexity Level
2b	Count and display odd numbers in an array	Array iteration, modulus operator	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
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
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
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
25	Program with command-line arguments for volume calculation	Macro usage, CLI argument parsing	Intermediate
32	Add two distances (feet/inches) using structures	Structure operations, type conversion	Intermediate

Q.No	Program Description	Key Concepts Tested	Complexity Level
33	Product management system with functions	Structure I/O, modular design	Advanced
34	Book structure demonstration	Structure declaration/initialization	Beginner
38	Employee management system with arrays of structures	Array of structures, batch processing	Intermediate
39	Student mark analysis with structures	Structure arrays, average calculation	Intermediate
40	Array of structures for student data	Structure array manipulation	Intermediate

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]) {
                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. Matrix Transpose (Q20)

```
#include <stdio.h>
#define SIZE 3
void transpose(int mat[SIZE][SIZE]) {
    for(int i=0; i<SIZE; i++) {
        for(int j=i+1; j<SIZE; j++) {
            int temp = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = temp;
        }
    }
```

```
}
int main() {
    int mat[SIZE][SIZE] = {{1,2,3},{4,5,6},{7,8,9}};
    transpose(mat);
    printf("Transposed matrix:\n");
    for(int i=0; i<SIZE; i++) {
        for(int j=0; j<SIZE; j++) printf("%d ", mat[i][j]);
        printf("\n");
    }
    return 0;
}
```

3. Recursive Fibonacci (Q14)

```
#include <stdio.h>
int fib(int n) {
    if(n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
int main() {
    int terms = 10;
    printf("Fibonacci series: ");
    for(int i=0; i<terms; i++)
        printf("%d ", fib(i));
    return 0;
}
```

Theory Q&A

1. Function Prototype vs Definition

Feature	Prototype	Definition
Body	No implementation	Full implementation
Purpose	Declaration before use	Actual functionality
Example	`int add(int, int);`	`int add(int a, int b) {return a+b}`

2. Recursion vs Iteration

Feature	Recursion	Iteration
Memory	Uses stack memory	Uses loop variables
Speed	Slower (function call overhead)	Faster
Use Case	Tree traversal, Fibonacci	Simple loops, array processing

3. Storage Classes

Class	Scope	Lifetime	Initial Value
`auto`	Local	Function call	Garbage
`static`	Local/Global	Program	Zero
`extern`	Global	Program	Zero
`register`	Local	Function call	Garbage

4. Structures vs Unions

Feature	Structure	Union
Memory	Allocates all members	Shares memory for members
Access	All members accessible	Only one member at a time
Size	Sum of member sizes	Size of largest member

Key Notes

- **Command-line arguments** use `argc` and `argv[]` parameters in `main()`
- **Macros** use `#define` for text substitution without type checking
- **Structure arrays** allow managing multiple records efficiently
- **Matrix operations** require nested loops for 2D array traversal

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

⁂