DS Laboratory assignment I:

Instructions:

Proper documentation is crucial in programming as it enhances code readability, maintainability, and usability, especially in collaborative environments or when the codebase is revisited after some time. Here are some guidelines you can suggest to students on how to effectively document their programs, particularly when using C:

Effective Documentation Practices for Programming in C

1. Use Comments Wisely

- Inline Comments: Use these sparingly to clarify complex parts of your code. Place them on the same line as the code or directly above the code block they explain.
- Block Comments: At the beginning of each file and function, use block comments to provide an overview of what the code does, including details about the purpose, parameters, return values, and any exceptions handled.

2. Documenting Functions

- Every function should have a comment block at the start that explains:
  - The purpose of the function.
  - Parameters it accepts, with data types and a brief description of each.
  - The return value, with its data type and what it represents.
  - Any side effects or exceptions the function may cause or handle.

3. Use Meaningful Names

- Choose variable and function names that convey their purpose without requiring additional comments. For example, `calculateTotal()` is more descriptive than `doCalc()`.

4. Maintain Readable Code Structure

- Organize your code with proper indentation and consistent style conventions. This includes using spaces and line breaks to separate logical blocks of code, making it easier to follow.

5. Include a Header Comment

- At the top of each file, include a comment block that describes:
  - The file's contents.
  - The author(s) and the date of creation.
  - Software license information if applicable.
  - A revision history or a link to a version control system where the history can be tracked.

6. Explain Magic Numbers

- Replace magic numbers with named constants, and use comments to explain why these constants are needed. For example, instead of just using `3.14159`, define `const double PI = 3.14159;` with a comment about the usage.

7. Regular Updates

- Update comments as you update your code. Outdated comments can be more misleading than no comments at all.

8. Include Examples

- In complex algorithms or functions, a few lines of example usage or expected input/output can be very helpful.

9. Documentation Tools

- Encourage the use of tools like Doxygen for C, which can help create an HTML or PDF documentation from annotated source code. This is particularly useful for large projects.

10. Review Documentation

- Just as you review code, review documentation for clarity, accuracy, and completeness, ideally with peers or mentors.

DS Laboratory assignment I:

Additional Advice:
- Stress the importance of balancing the amount of comments; too few comments can leave code cryptic, while too many can clutter the code and distract from its logic.
- Encourage practicing good documentation habits from the start—it's easier than trying to document a whole project at the end.

Assignment I: Programming Assignment Questions on Arrays and Matrices in C

1. Reverse Array: Write a program to reverse an array of integers. The user should not input the number of elements and the elements themselves, and your program should output the reversed array.

2. Matrix Transposition: Create a program that transposes a matrix. The program should generate the random matrix as the input and output the transposed matrix.

3. Check Symmetric Matrix: Write a program that checks if a square matrix is symmetric. A matrix is symmetric if it is equal to its transpose.

4. Sparse Matrix Representation: Implement a program that converts a matrix into its sparse matrix representation and displays the result. The sparse representation should list out only the non-zero elements along with their row and column indices.

5. Diagonal Sum: Develop a program that calculates the sum of the main diagonal elements of a square matrix. Generate the random matrix of different data type to be given as input.

6. Row and Column Sum: Write a program that calculates the sum of each row and each column of a matrix. The program should display the sums separately for rows and columns.

7. Rotate Matrix by 90 Degrees: Create a program that rotates a square matrix by 90 degrees clockwise. The user should input the matrix, and the output should be the rotated matrix.

8. Multiplication of Two Matrices: Implement a program that multiplies two matrices. The user inputs the dimensions and elements of both matrices. Your program should output the resulting matrix.

9. Dynamic Array Operations: Write a program that performs operations like insert, delete, and search on an array. The size of the array should dynamically increase or decrease as elements are added or removed.

10. Pascal's Triangle Using Arrays: Develop a program that generates Pascal's triangle up to a given number of rows. The number of rows should be input by the user, and the output should display the triangle.

---------- **Good luck** ----------