# POINTO CODING ASSIGNMENT

This assignment consists of two problem statements, and candidates are required to submit their final code in GitHub repositories. The main goal of the assignment is to assess the approach and problem-solving strategies used by candidates. It is essential that the code be written with production-quality standards in mind.

## Guidelines for the Assignment:

1. **Code Quality:** Ensure your code is clean, readable, and well-structured. Use meaningful variable names, and add comments where necessary to explain complex logic.

2. **Modularization:** Break your code into smaller, reusable functions or classes. Each function should ideally have a single responsibility.

3. **Error Handling:** Proper error handling is a must. Ensure that your code can handle edge cases and unexpected inputs gracefully, and provide clear error messages.

4. **Testing:** Write unit tests to validate your code. Cover all major edge cases and ensure your code behaves as expected in different scenarios. If possible, write integration tests to verify how different components work together.

5. **Version Control:** Use GitHub for version control. Commit your changes regularly with meaningful commit messages that explain the changes made. Follow a branching strategy (e.g., feature branches for new features, main for stable code).

6. **Performance Considerations:** Pay attention to the performance of your solution. While correctness is the primary focus, ensure that your solution is efficient and can scale with large inputs.

7. **Documentation:** Include clear and concise documentation. This should explain the purpose of the code, how to set up the environment, and how to use the solution. Include any necessary installation steps and dependencies.

8. **Readability:** Prioritize code readability over brevity. Ensure your code is easy for someone else to understand, even if they're unfamiliar with the problem you're solving.

9. **Security:** If your solution involves user input, be sure to include security measures (e.g., validation/sanitization) to prevent vulnerabilities like injection attacks.

By following these guidelines, you will demonstrate not only technical ability but also a solid understanding of best practices in software development.

# Problem Statement 1

The local commuter railroad operates between several cities in India, with each route being one-way due to budget constraints. This means that a route from Delhi to Chennai does not automatically imply a return route from Chennai to Delhi. In fact, even if both routes do exist, they could be distinct and have different distances.

The goal of this problem is to assist the railroad in providing valuable route information to its customers. Specifically, you will need to calculate the following:

- The total distance along a specific route,
- The number of different routes available between two cities,
- The shortest possible route between two cities.

**Input**

A directed graph where a node represents a city and an edge represents a route between two cities. The weighting of the edge represents the distance between the two cities. A given route will never appear more than once, and for a given route, the starting and ending city will not be the same city.

**Output**

For test input 1 through 5, if no such route exists, output ‘NO SUCH ROUTE’. Otherwise, follow the route as given; do not make any extra stops! For example, the first problem means to start at city A, then travel directly to city B (a distance of 5), then directly to city C (a distance of 4).

1. The distance of the route A-B-C.
2. The distance of the route A-D.
3. The distance of the route A-D-C.
4. The distance of the route A-E-B-C-D.
5. The distance of the route A-E-D.
6. The number of trips starting at C and ending at C with a maximum of 3 stops. In the sample data below, there are two such trips: C-D-C (2 stops). and C-E-B-C (3 stops).
7. The number of trips starting at A and ending at C with exactly 4 stops. In the sample data below, there are three such trips: A to C (via B,C,D); A to C (via D,C,D); and A to C (via D,E,B).
8. The length of the shortest route (in terms of distance to travel) from A to C.
9. The length of the shortest route (in terms of distance to travel) from B to B.
10. The number of different routes from C to C with a distance of less than 30. In the sample data, the trips are: CDC, CEBC, CEBCDC, CDCEBC, CDEBC, CEBCEBC, CEBCEBCEBC.

**Test Input**

For the test input, the cities are named using the first few letters of the alphabet from A to D. A route between two cities (A to B) with a distance of 5 is represented as AB5.

Graph: AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7

**Expected Output**

Output #1: 9
Output #2: 5
Output #3: 13
Output #4: 22
Output #5: NO SUCH ROUTE
Output #6: 2
Output #7: 3
Output #8: 9
Output #9: 9
Output #10: 7

**Notes**

- There must be a way to supply the application with the input data via text file.
- The application must run and you should provide sufficient evidence that your solution is complete by, as a minimum, indicating that it works correctly against the supplied test data.
- The submission should be production quality and it can be done in any language.
- You may not use any external libraries to solve this problem, but you may use external libraries or tools for building or testing purposes. Specifically, you may use unit testing libraries or build tools available for your chosen language.

# Problem Statement 2

Build a simple task manager application that allows users to create, update and delete tasks.

**Requirements**

1. Backend:
    - Build a RESTful API to manage tasks (CRUD).
    - Implement task attributes: title, description, status (To Do, In Progress, Done), dueDate, etc.
    - Store tasks (in-memory or database).
2. Frontend:
    - Create a user interface using React or Angular to interact with the task management API.
    - Users should be able to:
        - View tasks (filter by status).
        - Create new tasks.
        - Update task status (mark as "In Progress" or "Done").
        - Delete tasks.
    - Any additional features are most welcome.

**Notes**

- You can choose any language of your choice to implement these.
- Add a READme file to help us understand what you have built, constraints and stretch of the features.
- Additionally, try to add a docker file if possible to help run that application at once. If not, at least provide steps in READme.