Problem: Student Grade Tracker

Problem Description:

You are tasked to create a program to track student grades in a class. Each student has the following details:

- **Name** (String): The student's full name.
- **Roll Number** (Integer): A unique identifier for the student.
- **Marks** (Integer): The marks obtained by the student (out of 100).

The program should allow the user to:

1. Add a student’s details (name, roll number, and marks).
2. Display all students along with their details and grades.
    - A student’s grade is determined as follows:
        - Marks ≥ 90: Grade A
        - Marks ≥ 80 and < 90: Grade B
        - Marks ≥ 70 and < 80: Grade C
        - Marks < 70: Grade D
3. Calculate the average marks of all students.

The problem requires students to solve it in two ways:

1. Using **procedural programming** with separate functions and lists.
2. Using **object-oriented programming** with a `Student` class and relevant methods.

Requirements:

Procedural Programming Approach:

Write separate functions and use lists to manage the data. The following functions should be implemented:

1. `add_student(names, roll_numbers, marks)` :
    - Takes three lists (for names, roll numbers, and marks) as arguments.
    - Adds a new student’s details to the lists.
2. `display_students(names, roll_numbers, marks)` :

- Takes the three lists as arguments.
- Prints all students' details, including their grades.

3. `calculate_average(marks)` :
    - Takes the list of marks as an argument.
    - Calculates and prints the average marks.

Object-Oriented Programming Approach:

Create a `Student` class to encapsulate the properties and behaviors of a student:

1. **Attributes**:
    - `name` : The student's name.
    - `roll_number` : The student's roll number.
    - `marks` : The student's marks.
2. **Methods**:
    - `__init__(self, name, roll_number, marks)` : Constructor to initialize a student.
    - `get_grade(self)` : Calculates and returns the student's grade.

Create a `GradeBook` class to manage the list of students:

1. **Attributes**:
    - `students` : A list to store `Student` objects.
2. **Methods**:
    - `add_student(self, name, roll_number, marks)` : Adds a new student to the grade book.
    - `display_students(self)` : Displays all students with their details and grades.
    - `calculate_average(self)` : Calculates and displays the average marks of all students.

Example:

Input (Procedural or OOP):

```
1. Add Student: Alice, 101, 85
2. Add Student: Bob, 102, 92
```

```
3. Display All Students
4. Calculate Average Marks
```

Output:

```
Name: Alice, Roll Number: 101, Marks: 85, Grade: B
Name: Bob, Roll Number: 102, Marks: 92, Grade: A

Average Marks: 88.5
```