Section: C1+C2 **Time Limit: 55 minutes**

Student Course Enrollment System

Problem Statement

You are tasked with building a simple **Student Course Enrollment System** where:

- There are multiple **courses** available.
- Students can **enroll in multiple courses**.
- The system tracks which **students are enrolled in which courses**.

Functionality

Initial Input

- The program should read the **number of courses** as a **command-line argument**.
- For each course:
  - Assign a unique **Course ID** (starting from 1).
  - Take input for:
    - **Course Name** (string)
    - **Seat Capacity** (integer)

User Menu (Loop)

The program should repeatedly show the following menu to the user:

```
Menu:
1. Enroll a New Student in a Course
2. Enroll Existing Student in a Course
3. Drop Student from a Course
4. Display All Courses with Enrolled Students
5. Exit
```

Option 1: Enroll a New Student in a Course

- Prompt for:
  - **Student ID** (integer)
  - **Student Name** (string)
  - **Course ID** to enroll in
- If the course has available seats, enroll the student.
- If the course is full, show an appropriate message.

Option 2: Enroll Student in a New Course

- Prompt for:
  - **Student ID**
  - **Course ID**
- If the student exists and the course has seats, enroll them.
- If the course is full or student is already enrolled, show an appropriate message.

Option 3: Drop Student from a Course

- Prompt for:
  - **Student ID**
  - **Course ID**
- If the student is enrolled in the course, drop them.
- If not enrolled, inform the user.

Option 4: Display All Courses with Enrolled Students

- For each course, print:
  - **Course ID**, **Course Name**, **Capacity**, **Current Enrollment Count**
  - List of students enrolled (ID and Name)
- If no students are enrolled in a course, print:

```
No students enrolled.
```

Option 5: Exit

- Terminate the program.

Constraints

- Maximum of **30 courses**.
- Maximum of **100 students** in total.
- A course cannot exceed its seat capacity.
- A student cannot enroll in the same course more than once.

Classes to Implement

You must design only the following classes:

- **Course**
  Represents a course in the system.
- **Student**
  Represents a student.
- **Main**
  Contains the `main` method, handles user input/output, and controls program flow. The **Main** class should **not store** information about courses or students directly.

Submission Guidelines

- Create a folder named by your ID.
- Move all `.java` files into this folder.
- Zip the folder and submit the zip file.