Problem Statement

Student and Course Records:

- Each student is uniquely identified by a **student ID** and has associated information, including:
  - **Name**: The student's name.
  - **Year of Study**: The year the student is currently in.
  - **Completed Courses**: A list of course codes representing courses that the student has successfully completed (passed) in the past.
  - **Enrolled Courses**: A list of course codes representing courses the student is currently enrolled in.
- Each course is uniquely identified by a **course code** and includes:
  - **Course Name**: The name of the course.
  - **Number of Credits**: The credits associated with the course.
  - **Capacity**: The maximum number of students that can enroll in the course.
  - **Prerequisites**: A list of course codes representing the courses that must be completed before a student can enroll in this course.
  - **Time Slot**: The designated time slot for the course (e.g., Slot A, Slot B).

In your class definitions, you can maintain additional member functions or variables that you feel is needed to achieve operations efficiently.

Any operations that would lead to errors (e.g., adding courses before adding its prerequisites – details below) should be **rejected silently without error messages**.

[PART 1: SETTING UP THE BASICS] Implement the following operations in the Course Registration System.

- **Add Student**: Using the input format mentioned below.
- **Add Course**: When adding a course, the system must check that all prerequisite courses exist before adding the course. If any prerequisite does not exist, the course addition should be rejected.
- **Enroll Student**: When enrolling a student to a course, the system must check:
  - If the student or the course does not exist, reject the enrollment.
  - If the student has completed all prerequisites for the course.
  - If the course has available capacity.
  - If there are no scheduling conflicts with the student's currently enrolled courses.

  Enrollment is done if all above criteria are met. If a student has already completed the course, they are allowed to enroll in it (provided all the above criteria are met). If the student is already enrolled in the course, reject the enrollment.
- **Print Course**: List enrolled students in the output format given below.

Input Format

- The first line will contain an integer **Q**, representing the number of operations to be performed.
- For each operation:
  - If it is `add_student`, the next lines will include:
    - Student ID, Name, and Year of Study separated by spaces.
    - The number of completed courses
    - A list of completed course codes separated by spaces.
  - If it is `add_course`, the next lines will include:
    - Course Code, Course Name, Number of Credits, Capacity, Time Slot.
    - The number of prerequisites.
    - A list of prerequisite course codes separated by spaces.
  - If it is `enroll`, the next line will include:
    - Student ID and Course Code separated by a space.
  - If it is `print`, the next line will include the course code.

To simplify parsing of input, you can assume that student and course names, time slots, Student ID, Course Code, etc., do not each have spaces within them.

Output Format

The output of the program will consist of the results of the `print` operations. Each print operation should output either the (space-separated) list of student IDs enrolled in the specified course (in the order in which they were enrolled) or `Invalid Course <CourseCode>` if the course does not exist.

Example

Sample Input:

```
7
add_student
S001 Ayon 2 2
C101 C102
add_course
C101 IntroToProgramming 3 30 D 0

add_course
C201 DataStructures 3 30 A 1
C101
add_course
C202 Algorithms 3 25 B 1
C201
enroll
S001 C201
print
C201
print
C203
```

Sample Output:

```
Enrolled students in C201:
S001
Invalid Course C203
```

2. [PART 2: ENHANCED FEATURES] Building on the Course Registration System developed in Part 1, you will enhance the system to:

   (a) **Maintain course waitlist** when enrolling a student to a course at full capacity (i.e., augment the `enroll` operation to automatically place the student on a waitlist for the course, if the course is at full capacity). Please note that the waitlist functions on a first come first serve (FCFS) basis. A student can be on multiple courses' waitlists at a time.

   (b) **Enable dropping a student from a course, followed by automatic processing of waitlist** (by adding a new operation called `drop` which removes a course from a student's list of enrolled courses and does waitlist processing as detailed next). When a student successfully drops a course, automatically enroll the first student on the waitlist for that course if he/she meets all enrollment criteria from Part 1. Otherwise if enrollment fails, remove from the waitlist and attempt the next student for enrollment. Repeat this process until the next eligible student on the waitlist is found or if the waitlist becomes empty.

   (c) **Check for cyclic dependency in course prerequisites** (if needed). Can a cyclic dependency be introduced when adding a course (via `add course` operation along with its checks as listed in Part 1)? Say Yes/No in the answer sheet, and justify briefly. If you answered Yes, change the `add course` code to check and reject any `add course` operation that introduces cyclic dependencies.

All other operations and their functionalities of the system remain the same as in Part 1.

Input Format

- The first line will contain an integer Q, representing the number of operations.
- For the operations from Part 1, follow the same input format as in Part 1. For the new operation `drop`, the next lines will include:
  - Student ID and Course Code.
  - If either the student or course does not exist, the system will reject the drop request and no waitlist processing is done.

Output Format

Same as in Part 1, i.e., the output will consist of the results of the `print` operation.

Example

Sample Input:

```
10
add_student
S001 Ayon 2 2
C101 C102
add_student
S002 Shankar 2 1
C101
add_course
C101 DataStructures 3 1 A 0

add_course
C201 Algorithms 3 2 B 1
C101
enroll
S001 C101
enroll
S002 C101
enroll
S002 C201
drop
S001 C101
print
C101
print
C201
```

Sample Output:

```
Enrolled students in C101:
S002
Enrolled students in C201:
S002
```