A. Moodle replacement

time limit per test: 1 second
memory limit per test: 256 megabytes
input: input.txt
output: output.txt

We all know how hard it is to deal with Moodle! Imagine some university wants to replace this software and your job is to implement a new version with the requested functionalities.

You are tasked with building a Course Management System that manages students, and their exams. Each exam can be taken by students and their grades will be stored in the system, and students may take different types of exams and their grades will be saved in the system.

Each exam can be one of two types:

- Written Exam (with a duration in minutes).
- Digital Exam (with the specific software required).

You need to implement **Structures** for **"Student"** and **"Exam"** and a **Union** for **"ExamInfo"** and an **Enum** for **"ExamType"** with following fields.

Struct Student:

- `student_id`: Integer representing the unique ID of the student.
- `name`: String representing the student's name.
- `faculty`: String representing the student's faculty.

Struct Exam:

- `exam_id`: Integer representing the unique ID of the exam.
- `exam_type`: Type of the exam (Written or Digital) represented by enum ExamType.
- `exam_info`: Union ExamInfo containing additional information about the exam (either duration or required software).

Struct ExamGrade:

- `exam_id`: Integer representing the unique ID of the exam.
- `student_id`: Integer representing the unique ID of the student.
- `grade`: Integer representing the student's grade in the exam.

Enum ExamType:

- `WRITTEN`: Exam is taken using traditional pen and paper method and you should store the exam duration in minutes in the ExamInfo.
- `DIGITAL`: Exam is taken using computers and the name of the specific software for exam should be stored in the ExamInfo.

Union ExamInfo:

- `duration`: Integer representing the duration of the written exam (in minutes).
- `software`: String representing the software required for the digital exam.

The designed system should be able to respond to the following queries.

- `ADD_STUDENT [student_id] [name] [faculty]`
- `ADD_EXAM [exam_id] [exam_type] [exam_info]`
- `ADD_GRADE [exam_id] [student_id] [grade]`
- `UPDATE_EXAM [exam_id] [new_exam_type] [new_exam_info]`
- `UPDATE_GRADE [exam_id] [student_id] [new_grade]`
- `SEARCH_STUDENT [student_id]`
- `SEARCH_GRADE [exam_id] [student_id]`
- `DELETE_STUDENT [student_id]`
- `LIST_ALL_STUDENTS`
- `END`

Important: You are guaranteed to get the correct structure of the queries. (Correct command name, correct number of the arguments and separated by a white spaces each, if any arguments).

Note that when performing DELETE_STUDENT, all the stored exam results for the specific student must be deleted alongside the student as well.

Input

You will get the input from a file named `input.txt`. The input consists of queries in each line (there is only one query in each line), and the arguments (if any) for each query are separated with a whitespace.

- String lengths:
  - `name`: (1 < `name` < 20)
  - `faculty`: (4 < `faculty` < 30)
  - `software`: (2 < `software` < 20)
- Numeric ranges:
  - `student_id`: (0 < `student_id` < 1000)
  - `exam_id`: (0 < `exam_id` < 500)
  - `grade`: (0 <= `grade` <= 100)
  - `duration`: (40 <= `duration` <= 180)

Note: Numeric input values are not guaranteed to be within valid ranges, so ensure that you validate the numeric inputs accordingly and return error messages if needed. The length of the string fields and their content are not guaranteed to be in the given length range and consist of upper and lower case English letters. You need to validate the input for the length and content.

Output

The output should be written to a file named `output.txt` and every answer should be in a separate line. The format for each query response is as follows:

In case of any types of the errors in queries, print the appropriate error message for the first error you encounter in the query according to the priorities in `Type of error(s)` section in each query description and keep on answering the rest of the queries and do not terminate the program until receiving the `END` commmand.

- **ADD_STUDENT:**
  - Success: "`Student:` *`student_id`* `added`" [Replace the ***student_id*** with the given student_id in the query]

  Type of errors:

  1. Existing `student_id`: "`Student:` *`student_id`* `already exists`" [Replace the ***student_id*** with the given student_id in the query]
  2. Invalid `student_id`: "`Invalid student id`"
  3. Invalid `name`: "`Invalid name`"
  4. Invalid `faculty`: "`Invalid faculty`"

- **ADD_EXAM:**
  - Success: "`Exam:` *`exam_id`* `added`" [Replace the ***exam_id*** with the given exam_id in the query]

  Type of errors:

  1. Existing `exam_id`: "`Exam:` *`exam_id`* `already exists`" [Replace the ***exam_id*** with the given exam_id in the query]
  2. Invalid `exam_id`: "`Invalid exam id`"
  3. Invalid `exam_type`: "`Invalid exam type`"
  4. Invalid `duration`: "`Invalid duration`" (Applicable only for WRITTEN exam)
  5. Invalid `software`: "`Invalid software`" (Applicable only for DIGITAL exam)

- **ADD_GRADE:**
  - Success: "`Grade` *`grade`* `added for the student:` *`student_id`*" [Replace the ***grade*** and ***student_id*** with the given grade and student_id in the query]

  Type of errors:

  1. Not existing `exam_id`: "`Exam not found`"
  2. Invalid `exam_id`: "`Invalid exam id`"
  3. Not existing `student_id`: "`Student not found`"
  4. Invalid `student_id`: "`Invalid student id`"
  5. Invalid `grade`: "`Invalid grade`"

- **SEARCH_STUDENT:**
  - Success: "`ID:` *`student_id`*`, Name:` *`name`*`, Faculty:` *`faculty`*" [Replace the ***student_id***, ***name*** and ***faculty*** with the given student_id, name and faculty in the query]

  Type of error:

  1. Not existing student_id: "`Student not found`"
  2. Invalid student_id: "`Invalid student id`

- **SEARCH_GRADE:**
  - Success: "`Exam:` *`exam_id`*`, Student:` *`student_id`*`, Name:` *`name`*`, Grade:` *`grade`*`, Type:` *`exam_type`*`, Info:` *`exam_info`*" [Replace the ***exam_id***, ***student_id***, ***name***, ***grade***, ***exam_type*** and ***exam_info*** with the given information in the query]

  Type of errors:

  1. Not existing `exam_id`: "`Exam not found`"
  2. Invalid `exam_id`: "`Invalid exam id`"
  3. Not existing `student_id`: "`Student not found`"
  4. Invalid `student_id`: "`Invalid student id`"

- **UPDATE_EXAM:**
  - Success: "`Exam:` *`exam_id`* `updated`" [Replace the ***exam_id*** with the given exam_id in the query]

  Type of errors:

  1. Invalid `exam_type`: "`Invalid exam type`"
  2. Invalid `duration`: "`Invalid duration`" (Applicable only for WRITTEN exam)
  3. Invalid `software`: "`Invalid software`" (Applicable only for DIGITAL exam)

  - It is guaranteed that `student_id` and `exam_id` will be valid and existing for this specific query.

- **UPDATE_GRADE:**
  - Success: "`Grade` *`grade`* `updated for the student:` *`student_id`*" [Replace the ***student_id*** and ***grade*** with the given student_id and grade in the query]

  Type of errors:

  1. Invalid grade: "`Invalid grade`"

  - It is guaranteed that `student_id` and `exam_id` will be valid and existing for this specific query.

- **LIST_ALL_STUDENTS:**
  - Each student's information must be printed in the same format as `SEARCH_STUDENT`, with each student's details on a separate line in the order they have been added.

- **DELETE_STUDENT:**
  - Success: "`Student:` *`student_id`* `deleted`" [Replace the ***student_id*** with the given student_id in the query]
  - When deleting a student, delete all of their associated exam results as well.
  - It is guaranteed that `student_id` will be valid and existing for this specific query.

- **END**
  - Exit the program without printing anything in the output and any remaining queries will not be considered. You are guaranteed the command `END`.

Note: You need to print a newline character "\n" after your output.