Digital Pakistan Speed Programing Competition Online Mock Contest

Instructions

- Do not open the booklet unless you are explicitly told to do so. You can only read these instructions below.
- If you have any question regarding the problems, seek a clarification from the judges using DOMJudge.
- Before submitting a run, make sure that it is executable via command line. For Java, it must be executable via "javac" and for GNU C++ via "g++". Java programmers need to remove any "package" statements and source code's file name must be the same as of main class. C++ programmers need to remove any getch() / system("pause") like statements.
- Do not attach input files while submitting a run, only submit/attach source code files, i.e., *.java or *.cpp or *.py.
- Language supported: C/C++, Java and Python3
- Source code file name should not contain white space or special characters.
- You must take input from Console i.e.: Standard Input Stream (stdin in C, cin in C++, System.in in Java, stdin in Python)
- You must print your output to Console i.e.: Standard Output Stream (stdout in C, cout in C++, System.out in Java)
- Please, don't create/open any file for input or output.
- Please strictly meet the output format requirements as described in problem statements, because your program will be auto judged by computer. Your output will be compared with judge's output byte-by-byte and not tolerate even a difference of single byte. So, be aware! **Pay special attention to spaces, commas, dots, newlines, decimal places, case sensitivity etc.**
- All your programs must meet the time constraint specified.
- The decision of judges will be absolutely final.

Problem 04: ICPC Event Management - Optimizing Unique Event Tracking

Time limit: 1 second

ICPC Event Management Company is responsible for organizing various events, including competitions, entertainment shows, parties, processions, dinners, and academic exams. To ensure better accountability and preparation, ICPC needs a robust event tracking system. Each event is categorized by its type and subcategories, allowing for streamlined organization and efficient logistics planning. However, event codes sometimes contain repetitive patterns or **erroneous entries**, making it difficult to track events effectively.
To optimize the system, ICPC wants to determine the longest sequence of unique event codes from a given event schedule. If an **invalid entry** is found, the system should identify the error and flag it for correction.
Additionally, ICPC wants to know the exact **number of instances** of each event category type in the longest sequence to help in better event planning.

Event Categories and Subcategories:

ICPC organizes **7 different types of events**, each with multiple subcategories:

Category	Event Type	Subcategories (Example Codes)
A	Competitions	A01 (Local Contest), A02 (Regional Contest), A03 (National Contest), A04 (World Finals)
B	Entertainment	B01 (Concert), B02 (Movie Night), B03 (Stand-up Comedy), B04 (Theater)
C	Social Gatherings	C01 (Welcome Party), C02 (Networking), C03 (Farewell), C04 (Award Ceremony)
D	Dinners	D01 (Gala Dinner), D02 (Sponsor Dinner), D03 (Corporate Dinner), D04 (Team Dinner)
E	Processions	E01 (Opening Ceremony), E02 (Closing Ceremony), E03 (Cultural Parade), E04 (Community Walk)
F	Training Workshops	F01 (Algorithm Training), F02 (Competitive Coding), F03 (AI & ML Workshop), F04 (Career Guidance)
G	Exams	G01 (Preliminary Round), G02 (Elimination Round), G03 (Final Round), G04 (Certification Exam)

Each event and sub-event are represented by its **category letter** and a **two-digit number** indicating its subcategory. For example, G01 represents **Preliminary Round**, B02 represents **Movie Night**, and D04 represents **Team Dinner**.
ICPC wants to determine the longest contiguous sequence of unique event codes in a given event schedule. If there are multiple longest sequences of the same length, the one that starts with the **smallest event alphabetically** should be chosen.
Additionally, **if an invalid event code is detected**, the program should output -1 followed by the first erroneous entry.
The output should include:

1. **An integer** representing the number of distinct events in the longest unique sequence (or -1 if an error is found).
2. **The list of event codes** in that sequence, space-separated (or the invalid code if an error is found).
3. **The count of each event type category** in the sequence (e.g., 3 Competitions).

Input

- The first line contains the number of test cases.
- For each test case a single string *S*, where 0 < |S| < 1001, with |S| representing the total number of events.

Output

- Each line corresponds to the output of one test case, in the same order as the input. i.e.
    - If all event codes are valid:

- An integer representing the number of distinct events in the longest unique event sequence.
- The longest unique sequence of event codes.
- The exact number of instances per event type in the sequence.

o If an invalid event code is found:

- -1 followed by the first incorrect entry of length 3 where possible.

Sample Input	**Sample Output**
5 A01B02C03D04 A01A02A03 G01G02G03G04 A01A01A02A01 A01A02A3D02F09	4 A01 B02 C03 D04 1 Competitions 1 Entertainment 1 Social Gatherings 1 Dinners 3 A01 A02 A03 3 Competitions 4 G01 G02 G03 G04 4 Exams 2 A01 A02 2 Competitions -1 A3D