Parsing partitions

I recently discovered a trove of old text files that I seem to have been using when studying problems related to partitions of an integer. Unfortunately, I don't seem to have been very careful in saving those files, so I need a program to identify the correct ones, flag problematic ones, and output them in a standard format.

A *partition* of a positive integer n is a non-increasing sequence whose sum is n. For example, the partitions of 5 are (5), $(4,1)$, $(3,2)$, $(3,1,1)$, $(2,2,1)$, $(2,1,1,1)$ and $(1,1,1,1,1)$.

In my input files I can identify the following (loosely defined) format:

- There are four types of lines: comments, empty lines, partitions, and separators.
- Comment lines begin with the character `#`.
- Empty lines contain no characters, or just whitespace.
- Partition lines contain a sequence of positive integers separated by commas or by whitespace (but not both).
- Separator lines contain (only) a sequence of at least 3 hyphens.

The input files contain one or more *scenarios* delimited by the separator lines. A scenario must contain at least one partition.

There is a *standard* format which has the following additional properties:

- Partitions are written in non-increasing order, with the integers separated by single spaces.
- Separator lines consist of exactly eight hyphens.
- Empty lines contain just a single newline character (and no other whitespace).

Task

Write a program that takes input from `stdin` a data file. The objective is to write to `stdout` a standardised version of the input file.

In the event that the input file contains some invalid lines these should be written to output as comments of the form:

```
# INVALID: <line>
```

In the event that a scenario contains no valid partitions, the output for that scenario should begin with a comment of the form:

```
# INVALID SCENARIO
```

Otherwise, the output should simply echo comment lines, replace partitions and separators by their standard forms and replace multiple consecutive empty lines by single empty lines.

Further remarks

- A file may begin with a separator line. This is to be ignored in that case, i.e., the first scenario begins after that line.
- Similarly, there may or may not be a separator line at the end of the file.
- In standard form there is no separator line at the beginning or the end of the file.
- Any scenario consisting entirely of empty lines (or none at all, e.g., two consecutive separator lines) is just to be ignored in producing the standard form, i.e., these should not be marked as invalid but they should also not appear in the output (see sample I/O for an example).

Standards

For an achieved standard the program must operate as specified.

Merit criteria include well-structured and readable code, and extending the program to include helpful messages (and possibly suggestions) when invalid input is detected. In the latter case, it should be possible to run the program in either basic mode (as described above) or verbose mode (with additional output).

There are no clear excellence criteria for this étude.

Objectives

1.1, 1.4, 2.2, 3.5

(Individual)