The Problem

We need a program that can take a file containing an array of flat, non-hierarchical JSON objects, convert it into CSV format, and write it to disk.

The script should be able to translate this JSON file into a CSV file using the following rules:

- Each JSON object gets its own row in the CSV.
- Each key in the JSON object should have its value written to the corresponding column in the CSV
- Keys that are in multiple JSON objects should only have one corresponding column created in the CSV
- ***There is no known schema for the input JSON*** – the keys for each object can be anything.
- Column order is not important (yet).

The JSON objects themselves are composed of simple key-value pairs, where the values are either strings, integers, or booleans. We can assume that there are no nested JSON objects or arrays used as values. The keys themselves can vary between the different JSON objects contained in the array.

A sample JSON input file is as follows:

```
[
  {
    "id": 1,
    "first_name": "Jane",
    "last_name": "Doe"
  },
  {
    "id": 2,
    "first_name": "John",
    "middle_initial": "Q",
    "last_name": "Public",
    "birth_year": 1971
  },
  {
    "id": 3,
    "anonymous_user": true,
    "crm_id": "abc123"
  },
  {
    "id": 4,
    "first_name": "Albert",
    "last_name": "Einstein",
    "profession": "Scientist",
    "birth_year": 1879,
    "e_equals_mc_squared": true
  }
]
```

Note that each object within the array is flat, and contains keys that may or may not overlap with other objects in the array.

So for example, taking the example file up above, this script should generate the following output:

```
id,first_name,middle_initial,last_name,birth_year,anonymous_user,crm_id,profession,e_equals_mc_squared
1,Jane,,Doe,,,,,
2,John,Q,Public,1971,,,,
3,,,,,true,abc123,,
4,Albert,,Einstein,1879,,,Scientist,true
```

Notes

- You can use whatever language you want to implement the converter.
- Using a JSON parsing tool for the input file is fine. **We'd strongly prefer that you write the CSV itself out line-by-line, without using a CSV-writing library (or any native CSV-specific classes in whichever language you use).**
- The ordering of the columns does not need to be consistent from run to run.

- The input file can be consumed either from standard-in or from a file on disk – the choice is yours.
- Output should be written to a file on disk (as passed in via an argument)
- You can assume the input JSON content is valid without needing to verify.