ReversingLabs Written Python Development Test

This written test is part of ReversingLabs interview process. In order to promote a candidate to the next phase of the interview process the solution of the test should satisfy following criterias:

1. code readability and simplicity,
2. performance complexity understanding (big O notation),
3. completeness (no more, no less, than the task requires),
4. demonstrated and/or documented resource utilization awareness,
5. clean code awareness (like separation of concerns, modularity, validation, etc.),
6. production best-practices (like proper error and resource handling or external library usage).

Time taken to solve the written test is not evaluated and will not be reason for candidate disqualification. We hire all candidates who successfully pass interview process, so you are not competing with other candidates.

Task #1

Input data is contained in two disk files. Both files contain multiple entries separated by a newline character. The first file is of the following form:
`<first name> <ID number>`

The other file contains entries of the following format:
`<last name> <ID number>`

Write a program that, based on the information contained in input files, creates an output file with the format:
`<first name> <last name> <ID number>`

Example input:

```
Adam 1234
John 4321

Anderson 4321
Smith 1234
```

Expected output:

```
Adam Smith 1234
John Anderson 4321
```

Extension #1: sort output entries by the ID number
Extension #2: input data is too big to fit into main memory.

Task #2

Input data is a text containing three sorts of braces “()”, “{}”, “[]”. Write a program that will determine if braces in text are balanced.

Example input:

```
Python {is an easy to [learn]}, (powerful programming language. It)
has efficient high-level [(data structures) and a simple but
effective approach to object-oriented programming]. Python’s elegant
syntax and dynamic typing, together with its {interpreted nature,
make it an ideal language (for) scripting and rapid} application
development in many areas on most platforms.
```

Example output:

```
Braces are balanced.
```

Task #3

Write a decorator that stores the result of a function call and returns the cached version in subsequent calls (with the same parameters) for 5 minutes, or ten times -- whichever comes first.