Name:

|  |  |  |
| --- | --- | --- |
| Problem | Points | Score |
| Base Score | 50 |  |
| Level 1 | +10 |  |
| Level 2 | +10 |  |
| Level 3 | +10 |  |
| Level 4 | +10 |  |
| Level 5 | +10 |  |
| TOTAL | 100 |  |

Notes:

1. For this exam you are allowed to open a terminal window on your computer, you are allowed to web surf with Google, but you cannot use online chat or other interactive services.
2. As explained in class, this exam is structured so that you only get credit for a problem if you successfully pass the level you claim you have completed. If you apply for credit for a particular level and fail to meet the requirements of that level, you will receive no credit for that problem.
3. You cannot skip levels. For example, you must complete level 2 before you can apply for level 3.
4. Note that even though specific test cases are provided, your code must work for general cases. To receive full credit at a given level, you will need to pass previously unseen test cases.

**Deliverables:**

In this directory:

/data/courses/ece\_1111/2019\_fall/exams/exam\_04/<lastname\_firstname>

Create the following directories: l01, l02, l03, l04 and l05. In each of these directories, you will place a Python script named lXX.py (e.g., l02.py) that will be your solution for that level.

Program only in Python. You can use any Python resources available to you, but your code must run on nedc\_999. Do not use any libraries that are not installed. Once you complete a level, leave a working version of your code at that level, and copy your code to the next directory. This way you always have a working version of your code for the previous level.

There are two files you will use for testing:

/data/courses/ece\_1111/current/exams/exam\_04/picone\_joseph/l01.txt

/data/courses/ece\_1111/current/exams/exam\_04/picone\_joseph/l05.txt

Your programs should work for all similar files.

**L1:** Create a program, l01.py that reads a file and parses the input into a dictionary:

l01.py /data/courses/ece\_1111/current/exams/exam\_04/picone\_joseph/l01.txt

Loop over the dictionary and print out the key and the value. For this test case, your output should look like this:

Key: lincoln\_abraham Age: 37 State: illinois

Key: smith\_jones\_mary Age: 27 State: indiana

Key: ford\_henry. Age: 59 State: michigan

Key: edison\_thomas Age: 66 State: new\_jersey

Key: einstein\_albert Age: 88 State: new\_york

The keys must be done as lower case and join the first and last names with an underscore. You can assume the state name has no spaces in it (spaces have been replaced with underscores). State names should be printed as all lower case.

You can print the entries in any order, but they MUST be stored in a dictionary.

**L2:** Create a program, l02.py, that sorts the dictionary in numerical order and stores the result in a second dictionary. Print the result of the sorted dictionary:

l02.py /data/courses/ece\_1111/current/exams/exam\_04/picone\_joseph/l01.txt

Key: smith\_jones\_mary Age: 27 State: indiana

Key: lincoln\_abraham Age: 37 State: illinois

Key: ford\_henry Age: 59 State: michigan

Key: edison\_thomas Age: 66 State: new\_jersey

Key: einstein\_albert Age: 88 State: new\_york

Remember, your code needs to work for any file of this format. The dictionary must be sorted. You cannot simply print out the records in a sorted order.

**L3:** Create a program, l03.py, that locates the record that matches the input token and prints the corresponding entry. For example:

nedc\_999\_[1]]: l03.py <filename> abraham

Key: lincoln\_abraham Age: 37 State: illinois

nedc\_999\_[1]]: l03.py <filename> jones

\*> l03.py: no match for “jones” was found

Print out all records that match the input token. For example, an input token of “i” would display four of the five records in the example above.

**L4:** Create a program, l04.py, that produces the best match. For example, “abra” would clearly match the record indexed by “lincoln\_abraham”. But an input token of “hek” should produce a match with the record indexed by “ford\_henry” because it is the only record that contains “he”.

**L5:** Create a program, l05.py, that merges two files into a single sorted dictionary:

l05.py <file1> <file2>

The test case of l01.txt and l05.txt should produce this result:

nedc\_999\_[1]]: l05.py l01.txt l05.txt

Key: edison\_thomas Age: 66 State: new\_jersey

Key: einstein\_albert Age: 88 State: new\_york

Key: ford\_henry Age: 59 State: Michigan

Key: lincoln\_abraham Age: 37 State: illinois

Key: picone\_joseph Age: 99 State: illinois

Key: smith\_jones\_mary Age: 27 State: indiana

Key: zebra\_charlie Age: 01 State: pennsylvania

Sort the dictionary alphabetically.