CSCI 3104-Fall 2016: Programming Assignment # 2

Assigned date: Sunday 10/9/2016, **Due date:** Wednesday, 10/19/2016, online
Maximum Points: 40 points.
Note: This assignment *must be turned in online.*

This programming assignment requires you to implement two functions that are part of a server auditing application. Time is specified as an integer. Typically we specify time 0 to be a specific starting time eg., "Jan 1, 2016 at 12 midnight" and count the number of seconds elapsed since that time.

A server log is a list of N time intervals, wherein each interval (ℓ_i, u_i) specifies that some user was logged into the company server starting at time ℓ_i and ending at time u_i, with the starting/ending time points included.

$$[(\ell_1, u_1), (\ell_2, u_2), (\ell_3, u_3), \ldots, (\ell_N, u_N)]$$

You can assume that $\ell_i < u_i$, i.e, each user is logged in for more than 1 second.

P1 (20 points). Write a function `free_time_intervals(interval_lst, T)`, that takes in

1. list of time intervals `interval_lst`, as specified above
2. A floating point number `T` that specifies the ending time

It returns a list of all time intervals when no one was logged in:

$$[(a_1, b_1), \ldots, (a_k, b_k)]$$

such that

1. Each interval (a_i, b_i) is included in $[0, T]$, ie, $0 \leq a_i < b_i \leq T$, and
2. For each interval (a_i, b_i) no one is logged in during the times $a_i < t < b_i$. Note that the end points are exclusive: someone may have just logged out at a_i or b_i.
3. The intervals in this list should not overlap with each other.

Your algorithm should run in $\Theta(n \log(n))$ time.

Examples:

```
Input 1: [(5, 15)], T = 30
Output 1: [(0, 5), (15, 30)]

Input 2: [(1, 3), (2, 8), (3, 6), (2, 6), (10, 15), (12, 17), (19, 23), (27, 35)], T = 30
Output 2: [(0, 1), (8, 10), (17, 19), (23, 27)]

Input 3: [(5, 15), (18, 25), (3, 12), (4, 11), (1, 15), (18, 19)], T = 30
Output 3: [(0, 1), (15, 18), (25, 30)]
```

P2 (20 points) Write a function `max_logged_in(interval_lst, T)` returns a tuple `(max_logged_in_num, max_logged_in_time)`.

1. `max_logged_in_num` is a number that counts the maximum number of users logged in simultaneously between times 0 and T.
2. `max_logged_in_time` represents the the time instant between 0 and T when this maximum number was achieved. If multiple time instants had the same maximum number, return the earliest time.

Your algorithm should run in $\Theta(n \log(n))$ time.

Examples:

```
Input 1: [(5, 15)]
Output 1: (1, 5)

Input 2: [(1, 3), (2, 8), (3, 6), (2, 6), (10, 15), (12, 17), (19, 23), (27, 35)], T = 30
Output 2: (3, 2)

Input3: [(5, 15), (18, 25), (3, 12), (4, 11), (1, 15), (18, 19)], T = 30
Output 3: (4, 5)
```

Instructions: Download the zip file containing the files for this assignment. These include:

1. `pa2solution.py` : the file that you must complete and turn in online. Do not forget to fill in your name and acknowledge the declaration.
2. `pa2testscript.py`: the test script.
3. `test_1.txt,...,test_45.txt`: test files of various sizes. Some of the test files are huge. If your code fails these tests, you should try lots of smaller tests before you go to these. :-)

To run our test scripts:

```
  bash-3.2$ python3 pa2testscript.py test_1.txt
  Testing  test_1.txt
  Done.
     Test free_time_intervals Passed!
     Test max_logged_in passed!
  All test cases passed :-)
```

You may run multiple tests

```
  bash-3.2$ python3 pa2testscript.py test_1.txt test_2.txt test_3.txt test_4.txt
Testing  test_1.txt
Done.
   Test free_time_intervals Passed!
   Test max_logged_in passed!
Testing  test_2.txt
Done.
   Test free_time_intervals Passed!
   Test max_logged_in passed!
Testing  test_3.txt
Done.
   Test free_time_intervals Passed!
   Test max_logged_in passed!
Testing  test_4.txt
Done.
   Test free_time_intervals Passed!
   Test max_logged_in passed!
All test cases passed :-)
```

To run all from the terminal type: `python3 pa2testscript.py test_*.txt`
This will take about 5 minutes if you are code is efficient.