Collapsing Intervals (Due 11 Sep 2020)

An interval on the number line is denoted by a pair of values like so: (3, 8). Our intervals are closed. So this interval represents all numbers between 3 and 8 inclusive. The first number is always going to be strictly less than the second number. Normally in mathematics we represent a closed interval with square brackets. But in our program we will represent an interval by means of a tuple and tuples in Python are represented with parentheses.

If we have two intervals like (7, 12) and (4, 9), they overlap. We can collapse overlapping intervals into a single interval (4, 12). But the following two intervals (-10, -2) and (1, 5) are non-intersecting intervals and cannot be collapsed.

The aim in this assignment is take a set of intervals and collapse all the overlapping intervals and print the smallest set of non-intersecting intervals in ascending order of the lower end of the interval and then print the intervals in increasing order of the size of the interval.

Input: On HackerRank you will need to read from standard input like so:

```
$ python3 Intervals.py < intervals.in
```

The first line in *intervals.in* will be a single positive integer N ($1 \leq N \leq 100$). This number denotes the number of intervals. This will be followed by N lines of data. Each line will have two integer numbers denoting an interval. The first number will be strictly less than the second number. The intervals will not be in sorted order. Assume that the data file is correct. Here is the format of intervals.in

```
15
14 17
-8 -5
26 29
-20 -15
12 15
2 3
-10 -7
25 30
2 4
-21 -16
13 18
22 27
-6 -3
3 6
-25 -14
```

You will read in each pair of numbers and create a tuple out of them. You will store the tuples in a list. After you have read all the intervals and the list of tuples is complete, you will call the function *merge_tuples()*. This function will return a list of merged tuples in ascending order of the lower end of the tuples. For the input data file given the function *merge_tuples()* will return:

```
[(-25, -14), (-10, -3), (2, 6), (12, 18), (22, 30)]
```

You will take the output of the function *merge_tuples()* which is a list of tuples and pass it to the function *sort_by_interval_size()*. This function will return a list of tuples sorted by increasing order of the size of the intervals. If two intervals are of the same size then you should print the two intervals in ascending order of their lower ends. The list returned will be as follows:

```
[(2, 6), (12, 18), (-10, -3), (22, 30), (-25, -14)]
```

Output: You will print your output to standard out. Your output will have exactly two lines. The first line will be the output of the function *merge_tuples()* and the second line will be the output of the function *sort_by_interval_size()*. For the given input, the output will be as follows:

```
[(-25, -14), (-10, -3), (2, 6), (12, 18), (22, 30)]
[(2, 6), (12, 18), (-10, -3), (22, 30), (-25, -14)]
```

The file (Intervals.py) that you will be submitting will have the following structure. You may **NOT** change the names of the functions but you may add as many helper functions as needed. You will follow the standard coding conventions in Python.

```
# Input: tuples_list is an unsorted list of tuples denoting intervals
# Output: a list of merged tuples sorted by the lower number of the
#         interval
def merge_tuples (tuples_list):

# Input: tuples_list is a list of tuples of denoting intervals
# Output: a list of tuples sorted by ascending order of the size of
#         the interval
#         if two intervals have the size then it will sort by the
#         lower number in the interval
def sort_by_interval_size (tuples_list):

# Input: no input
# Output: a string denoting all test cases have passed
def test_cases ():
  assert merge_tuples([(1,2)]) == [(1,2)]
  # write your own test cases

  assert sort_by_interval_size([(1,3), (4,5)]) == [(4,5), (1,3)]
  # write your own test cases

  return "all test cases passed"

def main()
  # open file intervals.in and read the data and create a list of tuples

  # merge the list of tuples

  # sort the list of tuples according to the size of the interval

  # run your test cases
  '''
  print (test_cases())
  '''

  # open file intervals.out and write the output list of tuples
  # from the two functions

if __name__ == "__main__":
  main()
```

You can always add more functions than those listed.

For this assignment you may work with a partner. Both of you must read the paper on Pair Programming and abide by the ground rules as stated in that paper. If you are working with a partner then only one of you will submit the code. Make sure that in the header in HackerRank that you have your name and UT EID and your partner's name and UT EID. If you are working alone then you will just have your name and your UT EID.

Use the *HackerRank* platform to submit your code. We should receive your work by 11 PM on Friday, 11 Sep 2020. There will be substantial penalties if you do not adhere to the guidelines. HackerRank will not assign late penalties (if any), we will make the adjustments.

- Your code must run before submission.

- You should be submitting your file through HackerRank. HackerRank program. We will not accept files e-mailed to us.