Exercise 11.A: Mergesort with a Comparator

CS 1410

Background

The sort algorithms we have looked at in Module 8 have all sorted list elements in ascending order because they compare elements with the less-than operator. For example, in our **mergesort** program, the comparison appears as follows:

```
if L[i] < R[j]:
```

The effect of this comparison is that if **L[i]** is less than **R[i]**, then **L[i]** is considered to come before **R[i]** in the sorted result. Hence the *ascending* order of the output.

If we were to change to comparison to greater -han

```
if L[i] > R[j]:
```

which is the same as

```
if R[i] < L[j]:
```

then **R[i]** will precede **L[i]** in the result, yielding results in *descending* order.

In this exercise, you will change **mergesort** to accept two parameters: the list to sort, and a **comparator function**, which takes as parameters the two items **mergesort** is currently comparing. Your function will return **True** if the first parameter precedes the second in the desired ordering. By using a function as a parameter, you can sort lists by any desiring ordering, no matter how complex.

Data

The data to sort is in the file *data.csv* in the Exercise11A folder in Canvas. It is similar to *employees.csv* from Project 5, except the header row has been removed and an extra data record has been inserted. You will read this file into a list of tuples.

Requirements

Copy the code from *mergesort.py* to a new file named *msort.py*.

In *msort.py*, modify **mergesort** to take a second parameter, which will be a binary (two-arg) comparator function. In the updated **mergesort**, refer to that binary function whenever two items are being compared during the sorting. You only have to modify 4 lines of code.

You can verify that your function is working if the following code gives the output indicated below…

```
alist = [4,3,7,1,9]
mergesort(alist,lambda x,y: x > y)
print(alist)        # [9, 7, 4, 3, 1]
```

In another python module, *ex11a.py*, import **msort**, and read the data from *data.csv* into a **list of tuples**. Then call **mergesort** two times:

- First call **mergesort** with a comparator function that sorts the data in ascending order by **employee id**.
- Then call **mergesort** with a comparator that sorts the data in ascending order by **last name** as the primary key and **first name** as the secondary key.

After the first sort, write the results to the file *by_id.txt*. The first few lines should appear as…

```
('160769', 'Grant X. Mccarthy', '9846 Arcu. Road', 'Ketchikan', 'AK', '99634', '3', '1', '36025.38', '79.99', '11', '7117844-7', '288413-2164')
('163695', 'Beatrice P. Ware', '8903 Eget Rd.', 'Oklahoma City', 'OK', '20552', '1', '1', '46342.01', '86.47', '17', '26387832-9', '212804-3003')
('165966', 'Rooney Alvarado', '4963 Nisl. St. Ap #185', 'Gillette', 'WY', '20226', '3', '2', '34532.37', '21.53', '24', '37324307-8', '422046-0739')
('169927', 'Alexandra Y. Duke', 'P.O. Box 727', 'Springfield', 'MA', '50647', '2', '2', '31631.98', '95.73', '13', '23609043-4', '538878-8548')
('224568', 'Jenna B. Strickland', '70156 Rutrum Street', 'Phoenix', 'AZ', '86207', '1', '1', '57545.39', '24.70', '23', '33878326-4', '248847-6397')
…
```

After the second sort, write the results to the file *by_name.txt*. The first few lines should appear as…

```
('165966', 'Rooney Alvarado', '4963 Nisl. St. Ap #185', 'Gillette', 'WY', '20226', '3', '2', '34532.37', '21.53', '24', '37324307-8', '422046-0739')
('900100', 'Sopoline Bullock', '4963 Nisl. St. Ap #579', 'Idaho Falls', 'ID', '65094', '1', '2', '81929.49', '42.76', '16', '46979340-0', '485847-0083')
('983010', 'Jolene Burgess', 'P.O. Box 873', 'South Burlington', 'VT', '32036', '2', '2', '20042.77', '40.17', '23', '15300058-1', '828625-2906')
('283809', 'Hanae Dickson', '1633 Dolor Av.', 'Dover', 'DE', '61813', '1', '2', '51018.46', '91.22', '35', '48786143-K', '295643-0090')
('169927', 'Alexandra Y. Duke', 'P.O. Box 727', 'Springfield', 'MA', '50647', '2', '2', '31631.98', '95.73', '13', '23609043-4', '538878-8548')
```

Implementation Notes

Remember that to sort by composite keys, you compare tuples consisting of **(<primary_key>, <secondary_key>)**.

Submit your *msort.py*, *ex11a.py*, and two text files in a zip file on Canvas.