Merge Sort - Assignments

Download the zip file from https://data.gov.sg/dataset/number-of-mrt-lrt-stations (https://data.gov.sg/dataset/number-of-mrt-lrt-stations). In the zip file, find a csv file `number-of-mrt-and-lrt-stations.csv` which contains the year, number of MRT and LRT stations when there are new stations open. The data is already sorted by year.

Sample of the file.

```
year,mrt,lrt
2004,65,20
2005,65,31
```

1. Read Data from File

Implement a function `read_csv()` which reads above csv file and returns list of records.

- It takes in a parameter `file_path` which points to the csv file.
- It returns a nested-list of records. Each record is a list with the format of `[year, mrt_count, lrt_count]`, where `mrt_count` and `lrt_count` are integers.

In [4]:

```
import csv

def read_csv(file_path):
    with open(file_path) as f:
        reader = csv.reader(f)
        header = next(reader)
        data = [line for line in reader]
    return data
```

Test:

In [7]:

```
data = read_csv('./data/number-of-mrt-and-lrt-stations.csv')
print(data)
assert(data == [
    ['2004', '65', '20'],
    ['2005', '65', '31'],
    ['2006', '66', '31'],
    ['2007', '66', '33'],
    ['2008', '68', '33'],
    ['2009', '73', '33'],
    ['2010', '84', '33'],
    ['2011', '97', '34'],
    ['2012', '99', '34'],
    ['2013', '105', '35'],
    ['2014', '106', '38'],
    ['2017', '138', '42']
])
```

```
[['2004', '65', '20'], ['2005', '65', '31'], ['2006', '66', '31'], ['2007',
'66', '33'], ['2008', '68', '33'], ['2009', '73', '33'], ['2010', '84', '3
3'], ['2011', '97', '34'], ['2012', '99', '34'], ['2013', '105', '35'], ['20
14', '106', '38'], ['2017', '138', '42']]
```

2. Define Class `NewStations`

Define a class `NewStations` which contains information of the additional MRT and LRT stations over two consecutive published years.

- It contains 3 attributes, `period`, `added_mrt`, and `added_lrt`.
- Sample attribute values: `period = "2004-2005", added_mrt = 0, added_lrt = 11`.
- Implement its `__init__()` function to initialize its 3 attributes.
- Implement its `__str__()` function to print string in the format of `NewStations(2014-2015,0,11)`.

In [21]:

```
class NewStations:

    def __init__(self, period, added_mrt, added_lrt):
        self.period = period
        self.added_mrt = added_mrt
        self.added_lrt = added_lrt

    def __str__(self):
        return '{}({},{},{})'.format(self.__class__.__name__,
                                     self.period,
                                     self.added_mrt,
                                    self.added_lrt)

    def __lt__(self, other):
        return self.added_mrt < other.added_mrt

```

Test:

In [10]:

```
s = NewStations(period="2004-2005", added_mrt=0, added_lrt=11)
print(s)
assert(str(s) == 'NewStations(2004-2005,0,11)')
```

```
NewStations(2004-2005,0,11)
```

3. List of NewStations Objects

Implement a function `gen_newstations_list()` which takes in the output from `read_csv()` function, and returns a list of NewStations objects.

In [22]:

```
def gen_newstations_list(arr):

    result = []
    for i in range(len(arr)-1):
        period = '{}-{}'.format(arr[i][0], arr[i+1][0])
        added_mrt = int(arr[i+1][1]) - int(arr[i][1])
        added_lrt = int(arr[i+1][2]) - int(arr[i][2])
        obj = NewStations(period, added_mrt, added_lrt)
        result.append(obj)

    return result
```

Test:

In [23]:

```
newstations_list = gen_newstations_list(data)
print([str(x) for x in newstations_list])
assert([str(x) for x in newstations_list] == [
    'NewStations(2004-2005,0,11)',
    'NewStations(2005-2006,1,0)',
    'NewStations(2006-2007,0,2)',
    'NewStations(2007-2008,2,0)',
    'NewStations(2008-2009,5,0)',
    'NewStations(2009-2010,11,0)',
    'NewStations(2010-2011,13,1)',
    'NewStations(2011-2012,2,0)',
    'NewStations(2012-2013,6,1)',
    'NewStations(2013-2014,1,3)',
    'NewStations(2014-2017,32,4)'])
```

```
['NewStations(2004-2005,0,11)', 'NewStations(2005-2006,1,0)', 'NewStations(2
006-2007,0,2)', 'NewStations(2007-2008,2,0)', 'NewStations(2008-2009,5,0)',
'NewStations(2009-2010,11,0)', 'NewStations(2010-2011,13,1)', 'NewStations(2
011-2012,2,0)', 'NewStations(2012-2013,6,1)', 'NewStations(2013-2014,1,3)',
'NewStations(2014-2017,32,4)']
```

4. Merge Sort

Implement a function `sort_by_mrt()` which sorts the list by number of added MRT stations using Merge Sort algorithm.

- If you are implementing supporting function to merge sorted lists, name it as `merge_sorted_lists()` .

In [24]:

```
def merge_sorted_lists(arr1, arr2):
    result = []
    size1 = len(arr1)
    size2 = len(arr2)
    i,j = 0,0

    # when both arr1 and arr2 have items
    while i < size1 and j < size2:
#         if arr1[i].added_mrt < arr2[j].added_mrt:
        if arr1[i] < arr2[j]:
            result.append(arr1[i])
            i = i+1
        else:
            result.append(arr2[j])
            j = j+1

    # arr1 still has items, arr2 has no more item
    # arr1 has no item, arr2 still has items
    return result + arr1[i:] + arr2[j:]
```

In [25]:

```
def merge_sort(arr):
    if len(arr) <=1:
        return arr

    mid = len(arr)//2
    arr1 = merge_sort(arr[:mid])
    arr2 = merge_sort(arr[mid:])
    return merge_sorted_lists(arr1, arr2)
```

Test:

In [26]:

```
sorted_list = merge_sort(newstations_list)
print([str(x) for x in sorted_list])
assert([str(x) for x in sorted_list] == [
    'NewStations(2006-2007,0,2)',
    'NewStations(2004-2005,0,11)',
    'NewStations(2013-2014,1,3)',
    'NewStations(2005-2006,1,0)',
    'NewStations(2011-2012,2,0)',
    'NewStations(2007-2008,2,0)',
    'NewStations(2008-2009,5,0)',
    'NewStations(2012-2013,6,1)',
    'NewStations(2009-2010,11,0)',
    'NewStations(2010-2011,13,1)',
    'NewStations(2014-2017,32,4)'
])
```

```
['NewStations(2006-2007,0,2)', 'NewStations(2004-2005,0,11)', 'NewStations(2
013-2014,1,3)', 'NewStations(2005-2006,1,0)', 'NewStations(2011-2012,2,0)',
'NewStations(2007-2008,2,0)', 'NewStations(2008-2009,5,0)', 'NewStations(201
2-2013,6,1)', 'NewStations(2009-2010,11,0)', 'NewStations(2010-2011,13,1)',
'NewStations(2014-2017,32,4)']
```

5. Find the Median Number of New MRT Stations

Using value from the sorted list, find the median of the new MRT stations added in the list. Assign the value to `n`.

- If length of list is an odd number, the median is the middle value. If length of list is an even number, the median is the mean of the two middle values.

In [20]:

```
if len(sorted_list) % 2 == 1:
    mid = len(sorted_list)//2
    n = sorted_list[mid].added_mrt
else:
    mid = len(sorted_list)//2
    n = (sorted_list[mid].added_mrt + sorted_list[mid-1].added_mrt)/2

print(n)
```

```
2
```

Test:

In [97]:

```
assert(n==2)
```