CS6240
Homework 2
Name: Rushabh Shah

<u>GITHUB REPOSITORY</u>: https://github.ccs.neu.edu/rushabh0812/HW2

<u>Pseudo Codes</u>:

No-Combiner:

Mapper extracts station id, type of the record and temperature from value v

```
Method map (Key k, Value v):
        If (recordType == 'TMAX' or recordType == 'TMIN'):
                emit(stationID, (recordType, temperature, 1));

Method reduce(Key k,  value [(recordType1, temperature1,1),....... ]):
// key is the stationID
// value includes recordType, temperature and count

        maxCount =0
        minCount = 0
        maxSum = 0
        minSum = 0
        minAvg = 0
        maxAvg = 0

        for each record in value v:
                if( v.recordType == 'TMAX'):
                        maxSum += v.temperature
                        maxCount += 1
                if( v.recordType == 'TMIN'):
                        minSum += v.temperature
                        minCount += 1

        minAvg = minSum/minCount
        maxAvg = maxSum/maxCount

        emit(stationID, minAvg, maxAvg)
```

Combiner:

```
//Mapper extracts station id, type of the record and temperature from value v
Method map (Key k, Value v):
    If (recordType == 'TMAX' or recordType == 'TMIN'):
        emit(stationID, (recordType, temperature, 1));



Method combine(Key k,  value [(recordType1, temperature1,1),....... ]):
// key is the stationID
// value includes recordType, temperature and count

    maxCount =0
    minCount = 0
    maxSum = 0
    minSum = 0

    for each record in value v:
        if( v.recordType == 'TMAX'):
            maxSum += v.temperature
            maxCount += 1
        if( v.recordType == 'TMIN'):
            minSum += v.temperature
            minCount += 1


    emit(stationID, ("TMAX", maxSum))
    emit(stationID, ("TMIN", minSum))


Method reduce(Key k,  value [(recordType1, temperature1,1),....... ]):
// key is the stationID
// value includes recordType, temperature and count

    maxCount =0
    minCount = 0
    maxSum = 0
    minSum = 0
    minAvg = 0
    maxAvg = 0

    for each record in value v:
        if( v.recordType == 'TMAX'):
            maxSum += v.temperature
```

```
                maxCount += v.count
            if( v.recordType == 'TMIN'):
                minSum += v.temperature
                minCount += v.count

    minAvg = minSum/minCount
    maxAvg = maxSum/maxCount

    emit(stationID, minAvg, maxAvg)
```

In Mapper Combiner:

```
class map{
    initialize():
    // initialize hashmaps
    Hmax , Hmin

    method map(Key k, Value v){
        sum= 0
        total = 0
        // extract station id, type of the record and temperature from value v
        if(recordType = 'TMAX'):
            sum = Hmax{stationID}.temperature + temperature
            total = Hmax{stationID}.count + 1
            Hmax.add(stationID, (sum, total))

        if(recordType = 'TMIN'):
            sum = Hmin{stationID}.temperature + temperature
            total = Hmin{stationID}.count + 1
            Hmax.add(stationID, (sum, total))

    Method cleanup():

        for each key in Hmax:
            emit(key, (Hmax{key}.sum + Hmax{key}.count))

        for each key in Hmin:
            emit(key, (Hmin{key}.sum + Hmin{key}.count))
}
```

```
class reduce{

Method reduce(Key k,  value [(recordType1, temperature1,1),....... ]):
// key is the stationID
// value includes recordType, temperature and count

        maxCount =0
        minCount = 0
        maxSum = 0
        minSum = 0
        minAvg = 0
        maxAvg = 0

        for each record in value v:
                if( v.recordType == ‘TMAX’):
                        maxSum += v.temperature
                        maxCount += 1
                if( v.recordType == ‘TMIN’):
                        minSum += v.temperature
                        minCount += 1

        minAvg = minSum/minCount
        maxAvg = maxSum/maxCount

        emit(stationID, minAvg, maxAvg)

}
```

Secondary Sort:

```
// map function converts the value into a key and value pair . The key is an object of
CustomKey class
method map(Key k, value v) :
    from value v extract station-id, year, record-type and temperature
    if(record-type = "TMAX"):
        emit( (station-id,year), (year, 0,0,temperature,1))
    if(record-type= "TMIN") :
        emit((station-id,year),(year,temperature,1,0,0))

//The key is an object of type CustomKey. Custom key consists of two attributes.
stationId and year.
class CustomKey {
stationId
year
 method compareTo(Custom key k1 , Custom Key k2)
    compare the stationId's .
    if(same):
    compare year
}

//The partitioner takes the key which is of type CustomKey and returns an
appropriate partition based on stationId . All records with a particular station-id go
to same reducer
method partitioner(key):
    return hash(key.stationId)

//The combiner takes two parameters , key of the type CustomKey and list of values
having same key

method combiner(key, values[(year, maxSum0,maxCount0,
minSum0,minCount0),…]):
    maxSum=0
     minSum=0
    maxCount=0
    minCount=0

    for each v in values:
        maxSum += v.maxSum
        minSum += v.minSum
        maxCount += v.maxTempCount
        minCount += v.minCount

    emit(key, (year, maxSum, maxCount, minSum, minCount)
```

// The grouping comparator groups data by station id and all records having same stationId are sent to same reducer.

```
method customGroupComparator (Key k1, Key k2):
//Key consists of station-id and year
    compareValue = compare(k1.station-id, k2.station-id)
    return compareValue
```

//The reduce function takes two parameters. A key of type CustomKey and list of values having same key . Each value contains five components. year , TMIN sum , TMIN, count , TMAX sum, TMAX count. The values are received in the increasing order of year

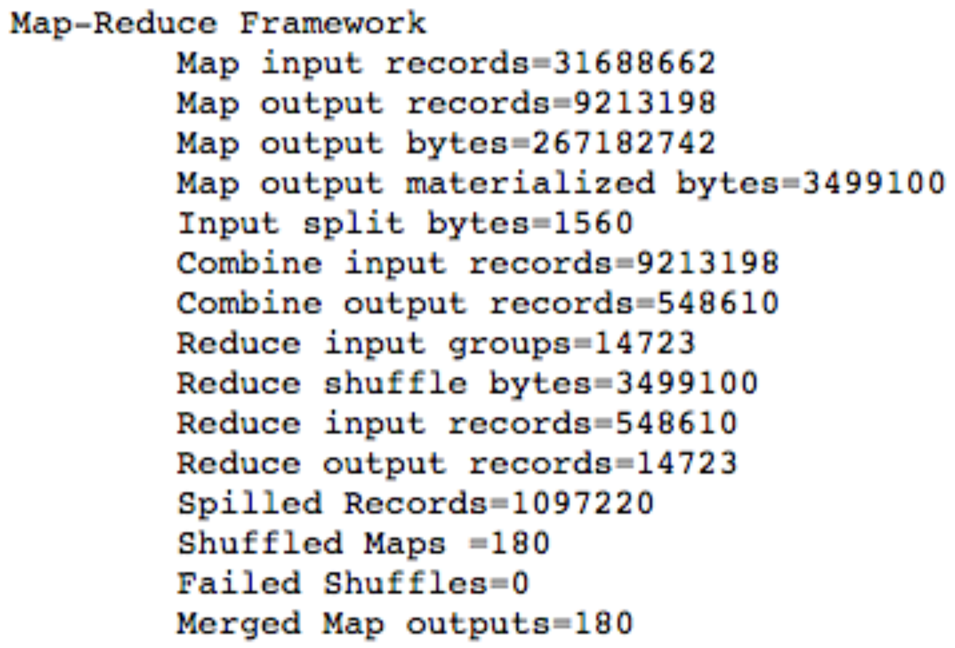
```
method reduce(key k, values[(year, minSum, maxSum, minCount, maxCount)] ):
    maxSum=0
     minSum=0
    maxCount=0
    minCount=0
    year = key.year

    for each in values:
        if(v.year is not equal to year):
            emit(key, (year, maxSum, maxCount, minSum, minCount)

        maxSum=0
        minSum=0
        maxCount=0
        minCount=0

    maxSum += v.maxSum
    minSum += v.minSum
    maxCount += v.maxTempCount
    minCount += v.minCount
```

The mapper emits records in the increasing order of stationId. This is achieved by overriding the inbuilt compareTo function. We defined our own custom compareTo function in CustomKey class . In this compareTo function , we first compare by stationId . If stationId's are equal, then we compare the years. Thus mapper emits records in increasing order of keys The grouping comparator the groups all the records having similar stationId and sends them to the reducer. Since we have defined our own custom comparator, the records in the reducer will be already present in the increasing order of year(In custom comparator , we compare by year when stationId's are equal). Thus we make use of map reduce's sorting ability to prevent explicit sorting of values in reducer. This in-turn eliminates the need for complex data structures needed to sort.

Running time of the programs:

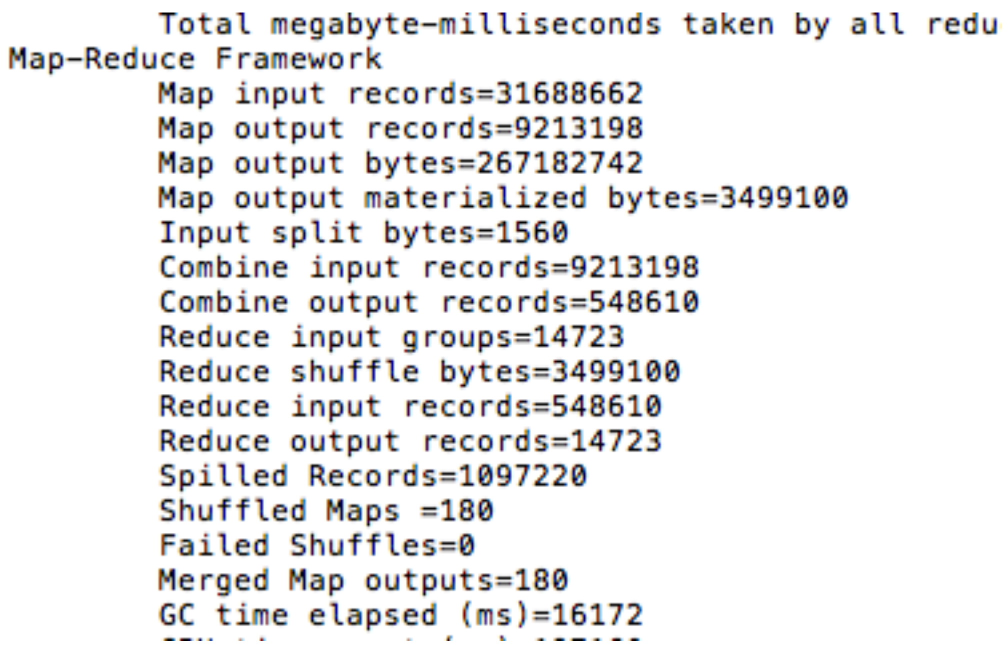
Program	Running Time 1 in seconds	Running Time 2 in seconds
No Combiner	108	104
Combiner	78	78
In Mapper Combiner	76	72
Secondary Sort	56	

Q. Was the Combiner called at all in program Combiner? Was it called more than once per Map task?
Ans: As we can see from the log files, the **Combiner was called** in the program for the two executions of the program Combiner.
First Execution:



```
Map-Reduce Framework
        Map input records=31688662
        Map output records=9213198
        Map output bytes=267182742
        Map output materialized bytes=3499100
        Input split bytes=1560
        Combine input records=9213198
        Combine output records=548610
        Reduce input groups=14723
        Reduce shuffle bytes=3499100
        Reduce input records=548610
        Reduce output records=14723
        Spilled Records=1097220
        Shuffled Maps =180
        Failed Shuffles=0
        Merged Map outputs=180
```

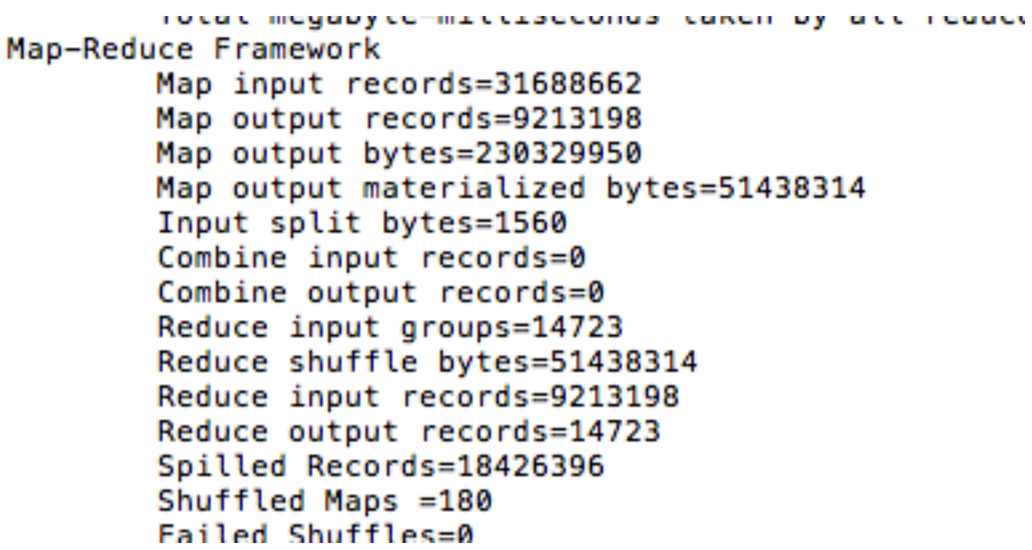
Second Execution:



```
          Total megabyte-milliseconds taken by all redu
  Map-Reduce Framework
          Map input records=31688662
          Map output records=9213198
          Map output bytes=267182742
          Map output materialized bytes=3499100
          Input split bytes=1560
          Combine input records=9213198
          Combine output records=548610
          Reduce input groups=14723
          Reduce shuffle bytes=3499100
          Reduce input records=548610
          Reduce output records=14723
          Spilled Records=1097220
          Shuffled Maps =180
          Failed Shuffles=0
          Merged Map outputs=180
          GC time elapsed (ms)=16172
          [illegible]
```

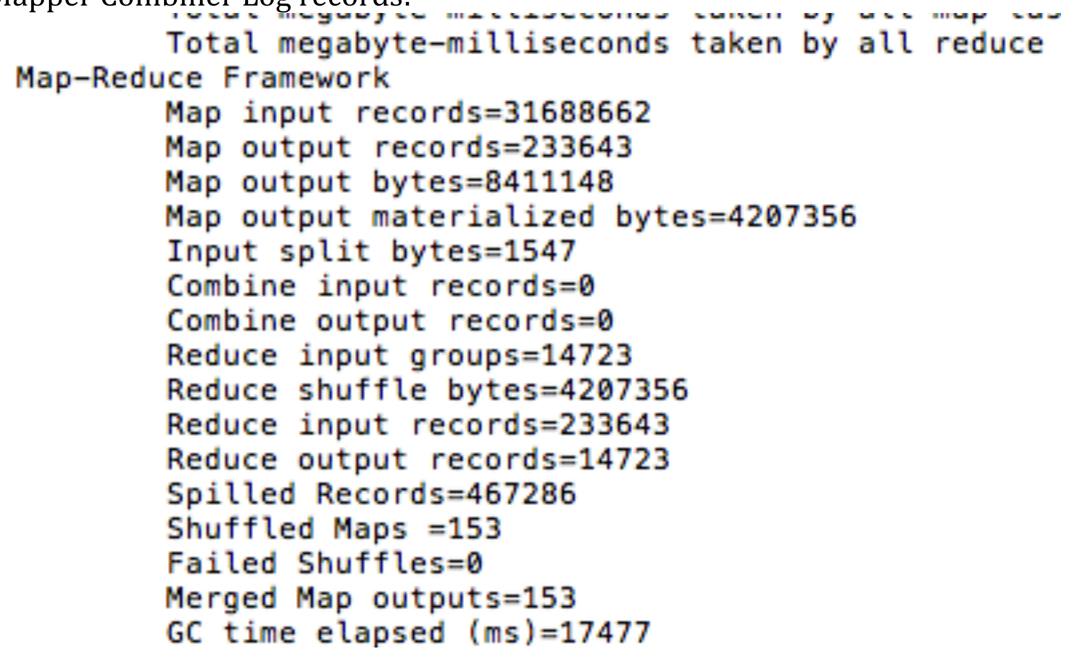
Q. Was the local aggregation effective in In-Mapper Combiner compared to No-Combiner?
Ans: **Yes**, the local aggregation in In-Mapper Combiner is effective compared to No-Combiner.

No Combiner Log records:



```
          Total megabyte-milliseconds taken by all reduc
  Map-Reduce Framework
          Map input records=31688662
          Map output records=9213198
          Map output bytes=230329950
          Map output materialized bytes=51438314
          Input split bytes=1560
          Combine input records=0
          Combine output records=0
          Reduce input groups=14723
          Reduce shuffle bytes=51438314
          Reduce input records=9213198
          Reduce output records=14723
          Spilled Records=18426396
          Shuffled Maps =180
          Failed Shuffles=0
```

In-Mapper Combiner Log records:



```
            Total megabyte-milliseconds taken by all map tas
            Total megabyte-milliseconds taken by all reduce
    Map-Reduce Framework
            Map input records=31688662
            Map output records=233643
            Map output bytes=8411148
            Map output materialized bytes=4207356
            Input split bytes=1547
            Combine input records=0
            Combine output records=0
            Reduce input groups=14723
            Reduce shuffle bytes=4207356
            Reduce input records=233643
            Reduce output records=14723
            Spilled Records=467286
            Shuffled Maps =153
            Failed Shuffles=0
            Merged Map outputs=153
            GC time elapsed (ms)=17477
```

By looking at the log records for In-Mapper Combiner and No-Combiner, we can see that there is a considerable reduction in the Map output records of both the programs.
The time taken to execute In-Mapper program is less than No Combiner as the less records are passed between the mapper and reducer.