Assignment 2

Stats Class: a case class for calculating min, max, mean and variance

```
case class Stats(var total: Int, var min: Int, var max: Int, var count: Int, var all_sizes: List[Int]) {
  def agg(other: Stats): Stats = Stats(
    total + other.total,
    math.min(min, other.min),
    math.max(max, other.max),
    count + other.count,
    all_sizes:+other.total
  )
  def gen_output(): String = {
    val mean = total.toDouble / count
    val variance = all_sizes.map(x=>math.pow(x.toDouble-mean,2)).sum / count
    return min+"B,"+max+"B,"+"%.0fB,%.0fB".format(math.floor(mean),math.floor(variance))
  }
}
```

Solution

1. Determine conversion constant for unit conversion.

```
val MB_to_B = 1048576
val KB_to_B = 1024
```

2. Create an RDD from input file and split each line by comma.

```
val file = sc.textFile(inputFilePath)
val lines = file.map(_.split(","))
```

3. Transform to pair of base URL and size of payload RDD. Then, convert size of payload to byte unit.

```
val pairs = lines.map(x => (x.head, x.last))
val bytes_pairs = pairs.map(x => if (x._2.takeRight(2) == "MB")
(x._1,x._2.dropRight(2).toInt*MB_to_B) else if (x._2.takeRight(2) == "KB")
(x._1,x._2.dropRight(2).toInt*KB_to_B)  else (x._1,x._2.dropRight(1).toInt))
```

4. Transform that RDD to a pair of base URL and stats object. Next, reduce key-value pair by key using aggregation function in stats class. Then, compute mean and variance by using *gen_output* function in stats class and map a key,value pair to one string output. Next, return a new RDD that has exactly one partition.

```
val result  = bytes_pairs.mapValues(f=>Stats(f,f,f,1,List(f))).reduceByKey(_ agg _)
val final_result  = result.map(x=>x._1+","+x._2.gen_output).repartition(1)
```

5. Finally, write an output.

```
final_result.saveAsTextFile(outputDirPath)
```