Adrien Protzel
3/5/2025

Homework: Spark Activities

1. Make a list of 25 integers across 3 partitions.
    a. t1 = sc.parallelize(range(1, 26), 3)
    b. t1.glom().collect()

```
>>> t1 = sc.parallelize(range(1, 26), 3)
>>> t1.glom().collect()
[[1, 2, 3, 4, 5, 6, 7, 8], [9, 10, 11, 12, 13, 14, 15, 16], [17, 18, 19, 20, 21, 22, 23, 24, 25]]
```

2. Make a list of 50 integers across 4 partitions, efficiently convert it to 2 partitions.
    a. t2 = sc.parallelize(range(1, 51), 4)
    b. t2.glom().collect()
    c. t2 = t2.coalesce(2)
    d. t2.glom().collect()

```
>>> t2 = sc.parallelize(range(1, 51), 4)
>>> t2.glom().collect()
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25], [26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37], [38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]]
>>> t2 = t2.coalesce(2)
>>> t2.glom().collect()
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25], [26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]]
```

3. Starting with a list of 26 integers 0 through 25 on 1 partition, end with a list of 26 integers split among two partitions, even numbers on one and odd on the other.
    a. t3 = sc.parallelize(range(0, 26), 1)
    b. t3.glom().collect()
    c. e = t3.filter(lambda x: x % 2 == 0)
    d. o = t3.filter(lambda x: x % 2 != 0)
    e. e.glom().collect()
    f. o.glom().collect()
    g. e = e.repartition(1)
    h. o = o.repartition(1)
    i. t3 = e.union(o).coalesce(2)
    j. t3.glom().collect()

```
>>> t3 = sc.parallelize(range(0, 26), 1)
>>> t3.glom().collect()
[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]]
>>> e = t3.filter(lambda x: x % 2 == 0)
>>> o = t3.filter(lambda x: x % 2 != 0)
>>> e.glom().collect()
[[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]]
>>> o.glom().collect()
[[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]]
>>> e = e.repartition(1)
>>> o = o.repartition(1)
>>> t3 = e.union(o).coalesce(2)
>>> t3.glom().collect()
[[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24], [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]]
```

4. Starting with 20 strings split somewhat evenly across 3 partitions, end with 4 partitions will ALL of the strings stored in one with the other 3 empty.
    a. t4 = sc.parallelize([f"string_{i}" for i in range(20)], 3)
    b. t4.glom().collect()
    c. t4 = t4.coalesce(1)
    d. t4 = t4.mapPartitionsWithIndex(lambda i, ii: [(0, list(ii))] if i == 0 else []).partitionBy(4).map(lambda x: x[1])
    e. t4.glom().collect()

```
>>> t4 = sc.parallelize([f"string_{i}" for i in range(20)], 3)
>>> t4.glom().collect()
[['string_0', 'string_1', 'string_2', 'string_3', 'string_4', 'string_5'], ['string_6', 'string_7', 'string_8', 'string_9', 'string_10', 'string_11'], ['string_12',
 'string_13', 'string_14', 'string_15', 'string_16', 'string_17', 'string_18', 'string_19']]
>>> t4 = t4.coalesce(1)
>>> t4 = t4.mapPartitionsWithIndex(lambda i, ii: [(0, list(ii))] if i == 0 else []).partitionBy(4).map(lambda x: x[1])
>>> t4.glom().collect()
[[['string_0', 'string_1', 'string_2', 'string_3', 'string_4', 'string_5', 'string_6', 'string_7', 'string_8', 'string_9', 'string_10', 'string_11', 'string_12', 's
tring_13', 'string_14', 'string_15', 'string_16', 'string_17', 'string_18', 'string_19']], [], [], []]
```

5. Compare the results of using repartition(20) directly on an RDD containing the values 0 through 99 with the results of first making a key value pair using the value as the key, then using partitionBy(20)
    a. t5 = sc.parallelize(range(100)).map(lambda x: (x, x))
    b. t5 = t5.repartition(20)
    c. t5 = t5.glom().map(len).collect()
    d. t5
    e. t5 = sc.parallelize(range(100)).map(lambda x: (x, x))
    f. t5 = t5.partitionBy(20)
    g. t5 = t5.glom().map(len).collect()
    h. t5

```
>>> t5 = sc.parallelize(range(100)).map(lambda x: (x, x))
>>> t5 = t5.repartition(20)
>>> t5 = t5.glom().map(len).collect()
>>> t5
[0, 0, 0, 0, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 0, 0, 0, 0, 0, 0]
>>> t5 = sc.parallelize(range(100)).map(lambda x: (x, x))
>>> t5 = t5.partitionBy(20)
>>> t5 = t5.glom().map(len).collect()
>>> t5
[5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
```

It would seem that repartition() randomly distributes the elements across 20 partitions, which result in half the partitions being empty. The partitionBy() on the other hand, evenly distributes elements across all partitions based.