Assignment 1

Task 1:

- Steps: Define the sequence of steps in the MapReduce task. In this case, there is a step with a mapper function and a reducer function;
- Mapper Function (mapper_get_age): It produces a key-value pair, where the key is the age and the value is the number of friends (converted into an integer);
- Reducer Function (reducer_average_ages): It yields a key-value pair where the key is the age, and the value is the mean (average) of the values (number of friends) associated with that age.
- **Result (extract):**

```
"49"    184.66666666666666
"50"    254.6
"51"    302.14285714285717
"52"    340.6363636363636
"53"    222.85714285714286
"36"    246.6
"37"    249.33333333333334
```

Task 2:

- Steps: Define the sequence of steps in the MapReduce task. In this case, there is one step with a mapper and a reducer function;
- Mapper Function (mapper_get_temp): It checks if the observation type is “TMIN” and, if so, yields a key-value pair where the key is the weather station and the value is the temperature (converted to an integer);
- Reducer Function (reducer_min_temp): It yields a key-value pair where the key is the weather station, and the value is the minimum temperature among the values associated with that weather station.
- **Result:**

```
"EZE00100082"   -135
"ITE00100554"   -148
```

Task 3:

- “*WORD_RE = re.compile(r"[a-zA-Z0-9_]+")*”: A regular expression used to find all the words in a character string. It matches alphanumeric characters and underscores;
- Steps: Define the sequence of steps in the MapReduce job. It consists of two steps: one for counting words and another for sorting and outputting the results. Each stage contains a mapper function and a reducer function;
- Counting words:
  - Mapper Function (mapper_get_words): It yields a key-value pair where the key is a lowercase word, and the value is 1;
  - Reducer Function (reducer_count_words): It yields a key-value pair where the key is a word, and the value is the sum of the counts (word frequency).
- Sorting and output of the results:
  - Mapper Function (mapper_make_counts_key): It takes the key and value resulting from the last phase as input and produces a key-value pair, where the key is 'None' and the value is a tuple of the word count (converted to integer) and the word;
  - Reducer Function (reducer_output_words): It receives a key and an iterator of word-count pairs. Sorts the word-count pairs based on their count and produces key-value pairs in which the key is a word and the value is its count.

Duarte Meneses 2019216949
Patricia Costa 2019213995

- **Result (extract):**

```
"the"   230
"and"   162
"a"     110
"to"    93
"he"    93
"his"   84
"she"   80
```

Task 4:

- Steps: Define the sequence of steps in the MapReduce job. It consists of two steps: one for getting the total amount spent by each customer and another for sorting and outputting the results. Each stage contains a mapper function and a reducer function;
- Getting the total amount spent by each customer:
  - Mapper Function (mapper_get_amount): It yields a key-value pair where the key is the customer ID, and the value is the amount spent (converted to a float);
  - Reducer Function (reducer_total_amount): It yields a key-value pair where the key is the customer ID, and the value is the sum of the amount spent by that customer.
- Sorting and output of the results:
  - Mapper Function (mapper_make_amounts_key): It takes the key and value resulting from the last phase as input and produces a key-value pair, where the key is 'None' and the value is a tuple of the total amount spent and the customer ID;
  - Reducer Function (reducer_output_amounts): It takes a key and an iterator of amount-customer pairs. It sorts the amount-customer pairs based on the amount and yields key-value pairs.
- **Result (extract):**

```
"68"    6375.449999999997
"73"    6206.199999999999
"39"    6193.109999999999
"54"    6065.389999999999
"71"    5995.660000000003
"2"     5994.59
```

Assignment 2

Task 1:

- Line 8: Division of RDDs into lists with the data we need (Weather Station, Observation Type, Temperature);
- Line 11: Filter only the lines that relate to the minimum temperature (as requested in the statement);
- Line 14: Convert the temperature to an integer, so that you can apply the min function;
- Line 17: For each capital (key), it returns only the lowest temperature value.
- **Result:**

```
ITE00100554     -148°C
EZE00100082     -135°C
```

Task 2:

- "*WORD_RE = re.compile(r"[a-zA-Z0-9_]+")*": A regular expression used to find all the words in a character string. It matches alphanumeric characters and underscores.
- Line 8: Dividing RDDs into words (by spaces);

- Line 11: Map the value of each word to 1;
- Line 14: It groups the words (keys) and adds up their occurrences (as the value of each word defaults to 1, the sum of these values corresponds to the total number of occurrences).
- **Result (extract):**

```
230     the
4       invisible
14      man
3       project
4       gutenberg
5       ebook
```

Task 3:

- Until Line 14: Same as Task 2;
- Line 17: Reverse the key with the value, from (word, value) to (value, word), so that we can sort the occurrences, which are the key;
- Line 20: Sort the occurrences of the words, which are the key.
- **Result (extract):**

```
230     the
162     and
110     a
93      he
93      to
84      his
80      she
```

Task 4:

- Line 8: Division of RDDs into lists with the data we need (Customer ID, Amount spent by customer);
- Line 11: Transform the amount spent by customers into a float, so that it can be added up;
- Line 13: It groups the customers (keys) and adds up the amount spent (values) for each.
- **Result (extract):**

```
44      4756.890
35      5155.420
2       5994.590
47      4316.300
29      5032.530
91      4642.260
70      5368.250
```

Task 5:

- Until Line 13: Same as Task 4;
- Line 15: Reverse the key with the value, from (Customer ID, Amount spent by customer) to (Amount spent by customer, Customer ID), so that we can sort the amount spent, these being the key;
- Line 17: Sort the amount customers spend, which is the key.
- **Result (extract):**

```
68      6375.450
73      6206.200
39      6193.110
54      6065.390
71      5995.660
2       5994.590
```

Task 6:

- Lines 10 and 12: Division of RDDs into lists with the data we need:
  - Marvel+Graph: Get the ID and the superhero name;
  - Marvel+Names: Each line represents a film and each ID represents a superhero who has appeared.
- Lines 14 and 15: Count the frequencies of each superhero ID;
- Line 17: Join the frequency RDD with the superhero names RDD using the superhero ID;
- Lines 19 and 21: Map the joined RDD to a new RDD with frequencies and superhero names, then sorts it in descending order by frequency.
- Lines 24 and 27: Retrieve the most popular frequency and filter superheroes with the same frequency.
- **Result:**

```
Most Popular Superhero: CAPTAIN AMERICA
Frequency: 1937
```

Task 7:

- Same as Task 6 but the sorting part is done in ascending order.
- **Result (extract):**

```
Least Popular Superhero: RED WOLF II
Frequency: 1

Least Popular Superhero: DEATHCHARGE
Frequency: 1

Least Popular Superhero: MARVEL BOY II/MARTIN
Frequency: 1
```

Assignment 3

Task 1:

- Line 8: Divide the read data into lists of what we need (Weather Station, Observation Type, Temperature);
- Line 11: Filter only the lines that relate to the minimum temperature;
- Line 14: Select only the weather station and temperatures from the three variables;
- Line 17: For each capital (grouped), return only the lowest temperature value.
- **Result:**

```
+--------------+--------------+
|WeatherStation|MinTemperature|
+--------------+--------------+
|   ITE00100554|          -148|
|   EZE00100082|          -135|
+--------------+--------------+
```

Task 2:

- Line 9: Replace non-alphanumeric characters with spaces;
- Line 12: Divide lines into words (by spaces);
- Line 15: Filtering the result from line 9 to remove any empty characters;
- Line 18: For each word (grouped), returns the number of occurrences (counting).
- **Result (extract):**

```
+------------+-----+
|        word|count|
+------------+-----+
|      online|    1|
|      biting|    1|
|       crest|    1|
|         few|    2|
|      deftly|    1|
|       still|    7|
```

Task 3:

- As in Task 2, but adding in line 21, sort the words in descending order according to their occurrences.
- **Result (extract):**

```
+----+-----+
|word|count|
+----+-----+
| the|  230|
| and|  162|
|   a|  110|
|  to|   93|
|  he|   93|
| his|   84|
| she|   80|
```

Task 4:

- Line 9: Divide the data read into lists of what we need (Customer ID, amount spent by customer);
- Line 12: For each customer (grouped), return the amount spent (summing).
- **Result (extract):**

```
+----------+------------------+
|CustomerID|  TotalAmountSpent|
+----------+------------------+
|        31|           4765.05|
|        85|           5503.43|
|        65|5140.3499999999985|
|        53| 4945.299999999999|
|        78| 4524.509999999999|
|        34|            5330.8|
|        81| 5112.709999999999|
```

Task 5:

- Up to Line 12: Same as Task 4;
- Line 15: Sort customers in descending order according to their expenses.
- **Result (extract):**

```
+----------+------------------+
|CustomerID|  TotalAmountSpent|
+----------+------------------+
|        68| 6375.449999999997|
|        73| 6206.199999999999|
|        39| 6193.109999999999|
|        54| 6065.389999999999|
|        71| 5995.660000000003|
|         2|            5994.59|
|        97| 5977.189999999995|
|        46| 5963.109999999999|
|        42| 5696.840000000003|
```

Task 6:

- Line 10: Lines from “Marvel+Graph” are split into words using spaces as delimiters;
- Line 13: For “Marvel+Names” extract superhero IDs and names;
- Line 16: Count how many times each superhero appears in “Marvel+Graph”;
- Line 19: Join frequency counts with superhero names using the IDs;
- Line 22: Sort the result by frequency in descending order;

- Lines 26 and 27: Retrieve the most popular frequency and filter superheroes with the same frequency.
- **Result (extract):**

```
Most Popular Superhero: "CAPTAIN AMERICA"
Frequency: 1937
```

Task 7:

- Same as Task 6 but the sorting part is done in ascending order
- **Result (extract):**

```
Least Popular Superhero: "BERSERKER II"
Frequency: 1

Least Popular Superhero: "BLARE/"
Frequency: 1

Least Popular Superhero: "MARVEL BOY II/MARTIN"
Frequency: 1
```

Assignment 4

Task 1:

- Line 12: Dividing Lines (streams) into words (by the spaces)
- Line 15: Group the words and count their occurrences
- Line 17: Sort the occurrences of words
- **Results (Input/Output):**

```
/assignment#4$ netcat -l 9999
pedro
iacd
teste
alunos
aulas
iacd
aulas
```

```
+------+-----+
|  word|count|
+------+-----+
|alunos|    1|
| teste|    1|
| pedro|    1|
| aulas|    2|
|  iacd|    2|
+------+-----+
```