Level 1

Question item 01

Input: Consider a transactional dataset of bought items in a supermarket. The dataset is stored in a collection of multi-line text files. Each line in a file is a key-value tuple of ⟨*key, item*⟩, where *key* is the identity of the transaction and *item* is an item bought in that transaction.

- The file contains digits, lowercase letters, white spaces, tabs, and newline characters.
- The key part and value part are separated by a single tab.

Sample input:

Files	file01	file02	file03
Content	t01 eggs t03 squid steak t02 milk bottle	t03 potato chips t03 chicken wings t01 eggs	t02 eggs t03 chicken wings t02 milk bottle

Main requirement: For each transaction identity, list all the distinct items and their associated counts. Furthermore, the list of elements [item, counts] must be sorted (in either ascending or descending order) in terms of counts; for ties, continue to sort the items in terms of dictionary order.

Output: Key-value tuples of (*key*, list of [*item, count*]), where *key* is the identity of a transaction, each pair of *[item, count]* represents the item bought in that transaction and the corresponding count.

- The key part and value part are separated by a single tab.

Sample output:

```
t01     [eggs, 2]
t02     [eggs, 1] [milk bottle, 2]
t03     [potato chips, 1] [squid steak, 1] [chicken wings, 2]
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 1

Question item 02

Input: Consider a transactional dataset of bought items in a supermarket. The dataset is stored in a collection of multi-line text files. Each line in a file is a key-value tuple of ⟨*key, item*⟩, where *key* is the identity of the transaction and *item* is an item bought in that transaction.

- The file contains digits, lowercase letters, asterisks, white spaces, tabs, and newline characters.
- The key part and value part are separated by a single tab.

Sample input:

Files	file01	file02	file03
Content	t01 eggs t03 squid steak t02 milk bottle*	t03 potato chips t03 chicken wings t01 eggs	t02 eggs t03 chicken wings* t02 milk bottle

Main requirement: Assume that each item costs X dollars, where X equals the number of letters in the item's name (except white spaces). Furthermore, we note that if the item's name is followed by an asterisk, its cost is 20% off. Calculate the total cost of each transaction.

- For example, "eggs" costs 4 dollars, and "red chili*" cost 8*0.8 = 0.64 dollars.

Output: Key-value tuples of (*key*, *amount*), where *key* is the identify of a transaction and *amount* is the total cost of all items in that transaction.

- The key part and value part are separated by a single tab.

Sample output:

t01	8	there are two "eggs"
t02	22	there are one "milk bottle", one "milk bottle*", and one "eggs"
t03	42.6	there are one "squid steak", one "potato chips", one "chicken wings", and one "chicken wings*"

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 1

Question item 03

Input: Consider a transactional dataset of bought items in a supermarket. The dataset is stored in a collection of multi-line text files. Each line in a file is a key-value tuple of ⟨*key, item*⟩, where *key* is the identity of the transaction and *item* is an item bought in that transaction.

- The file contains digits, lowercase letters, white spaces, tabs, and newline characters.
- The key part and value part are separated by a single tab.

Sample input:

Files	file01	file02	file03
Content	t01 eggs t03 squid steak t02 milk bottle	t03 potato chips t03 chicken wings t01 eggs	t02 eggs t03 chicken wings t02 milk bottle

Main requirement: An item is considered as "profitable" if it appears in *more than one transaction*. List all "profitable" items and the corresponding sets of transactions in which these items appear. *Note that you must not print out any item that is not "profitable".*

Output: Key-value tuples of (*item*, list of [*identity*]), where *item* is a profitable item and *identity* is the identity of the transaction that includes *item*.

- The key part and value part are separated by a single tab.

Sample output:

```
eggs    t01, t02
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 1

Question item 04

Input: Consider a dataset of 1-D points, which are clustered into groups. The dataset is stored in a collection of multi-line text files. Each line in a file is a key-value tuple of ⟨*group, point*⟩, where *group* is the identity of the group identity and *point* is the 1-D coordinate of a point belonging to that group.

- The file contains digits, lowercase letters, white spaces, tabs, and newline characters.
- Coordinate values are integers.
- The key part and value part are separated by a single tab.

Sample input:

Files	file01	file02	file03
Content	g01 5 g03 10 g02 7	g03 11 g03 1 g01 2	g02 6 g01 4 g02 3

Main requirement: Find the center and the centroid for each group of 1-D points.

- For a group of 1-D points, the center typically refers to the "mean" or "average" of the points, while the centroid is the middlemost point.

Output: Tuples of (*group*, *center, centroid*), where *group* is the identity of a group, *center* and *centroid* are the 1-D coordinates of the center and centroid of the group, respectively.

- The elements of a tuple are separated by a single tab.

Sample output:

```
g01	3.67	4
g02	5.33	6
g03	7.33	10
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 1

Question item 05

Input: Consider a dataset of 1-D points. The dataset is stored in a collection of multi-line text files. Each line in a file is a key-value tuple of ⟨*point, coordinate*⟩, where *point* is the name of a point and *coordinate* is the 1-D coordinate of that point.

- The file contains digits, lowercase letters, white spaces, tabs, and newline characters.
- Coordinate values are integers.
- The key part and value part are separated by a single tab.

Sample input:

Files	file01	file02	file03
Content	p1 5 p2 10 p3 7	p4 11 p5 1 p6 2	p7 6 p8 4 p9 3

Main requirement: There are three clusters of points, whose centers are defined in advance. Assign each point to the nearest cluster and calculate the new centers.

- In your source code, fix the number of clusters to 3 and arbitrarily choose the three centers.
- For a group of 1-D points, the center typically refers to the "mean" or "average" of the points.
- Ties are broken randomly. It is unnecessary to format the number.

Output: Three tuples of (*old center*, *new center,* list of [*point*]), one per cluster, where *old center* is the coordinate of the old center (defined in advance), *new center* is the coordinate of the new center, and *point* is the name of a point belonging to that cluster.

- The elements of a tuple are separated by a single tab.
- Two consecutive points in the list are separated by a whitespace.

Sample output: Assume that the three cluster centers are 2, 6, and 9.

```
2       2.5     p5 p6 p8 p9
6       6       p1 p3 p7
9       10.5    p2 p4
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 1

Question item 06

Input: Consider a dataset of 1-D points. The dataset is stored in a collection of multi-line text files. Each line in a file is a key-value tuple of ⟨*point, coordinate*⟩, where *point* is the name of a point and *coordinate* is the 1-D coordinate of that point.

- The file contains digits, lowercase letters, white spaces, tabs, and newline characters.
- Coordinate values are integers.
- The key part and value part are separated by a single tab.

Sample input:

Files	file01	file02	file03
Content	p1 5 p2 10 p3 7	p4 11 p5 1 p6 2	p7 6 p8 4 p9 3

Main requirement: Let the user enter a single value as the query point *q*. Group the points following their distances to the query point.

- The user enters the value via an argument in the command line.

Output: Key-value tuple of (*distance*, list of [*point*]), where *distance* is the absolute distance from a point to the query point, and *point* is the name of a point that is *distance* unit away from the query point.

- The key part and value part are separated by a single tab.
- Two consecutive points in the list are separated by a whitespace.

Sample output: Assume that the query point is 4.

```
0	p8
1	p9 p1
2	p6 p7
3	p5 p3
6	p2
7	p4
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 1

Question item 07

Input: Consider a digital single-channel image, whose pixels are integer numbers in range of [0, N]. The image is stored in a collection of multi-line text files. Each line in a file is list of ⟨*pixel*⟩, representing one row of the image, where *pixel* is a pixel in that row.

- The file contains digits, white spaces, tabs, and newline characters.

Sample input:

Files	file01	file02	file03
Content	1 2 3 4 4 3 2 1 5 6 7 8	8 7 6 5 1 3 5 7 7 5 3 1	2 4 6 8 8 6 4 2 1 4 2 5

Main requirement: Compute the global histogram of the image using N bins, one per pixel value.

- A histogram plots the distribution of a numeric variable's values as a series of bars. Each bar covers a range of numeric values called a bin or class; a bar's height indicates the frequency of data points with a value within the corresponding bin.

Output: Key-value tuple of (*value*, *count*), where *value* is the value of a pixel (in range of [0, N]), and *count* is the number of pixels having the same values *value*.

- The key part and value part are separated by a single tab.

Sample output:

```
0	0
1	5
2	5
3	4
4	5
5	5
6	4
7	4
8	4
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 1

Question item 08

Input: A Telecom company keeps records for its subscribers in a collection of multi-line text files. Each record is on a separate line and it follows the below format.

FromPhoneNumber|ToPhoneNumber|CallStartTime|CallEndTime|STDFlag

- A STD Call is a long distance call. If STD Flag is 1 that means it was as STD Call.

Sample input:

	Content
file01	9665128505|8983006310|2015-03-01 07:08:10|2015-03-01 08:12:15|0 9665128505|8983006310|2015-03-01 09:08:10|2015-03-01 09:12:15|1 9665128505|8983006310|2015-03-01 09:08:10|2015-03-01 10:12:15|0
file02	9665128506|8983006312|2015-03-01 09:08:10|2015-03-01 10:12:15|1 9665128507|8983006313|2015-03-01 09:08:10|2015-03-01 12:12:15|1 9665128505|8983006310|2015-03-01 11:08:10|2015-03-01 11:12:15|1

Main requirement: Find out all phone numbers who are making more than 60 mins of STD calls. By identifying such subscribers, the Telecom company wants to offer them STD (Long Distance) Pack which would efficient for them instead spending more money without that package.

Output: Key-value tuples of (*phone_number*, *duration_in_minutes*), where *phone_number* is the phone number and *duration_in_minutes* is the corresponding duration measured in minutes.

- The key part and value part are separated by a single tab.

Sample output:

```
9665128506	64
9665128507	184
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 2

Question item 09

Input: Consider a database of two tables, FoodPrice and FoodQuantity, whose records are stored together in a collection of multi-line text files. Each line in a file is a tuple of three elements, ⟨*table*, *key*, *value*⟩, where *table* is the name of the table (only FoodPrice or FoodQuantity), *key* is the common key of two tables, and *value* is the value at the corresponding record in FoodPrice or FoodQuantitty.

- The file contains digits, lowercase letters, white spaces, and newline characters.
- Two consecutive elements are separated by a whitespace. An item of type string is always a single word (no whitespaces in between).

Sample input:

Files	file01	file02
Content	FoodPrice Pizza 8 FoodPrice Burger 6 FoodQuantity Pizza 3	FoodPrice FrenchFries 4 FoodQuantity Burger 5 FoodQuantity Cake 2

Main requirement: Perform LEFT OUTER JOIN FoodPrice table to FoodQuantity table.

- A value of null is used to represent the fact that there is no matched value.

Output: Tuples of (*key*, *first_value, second_value*), where *key* is the common key of two tables, *first_value* and *second_value* are the corresponding values in the first table and the second table, respectively.

Sample output:

```
Pizza 8 3
Burger 6 5
FrenchFries 4 null
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.
- SQL joins must be implemented from scratch.

Level 2

Question item 10

Input: Consider a database of two tables, FoodPrice and FoodQuantity, whose records are stored together in a collection of multi-line text files. Each line in a file is a tuple of three elements, ⟨*table*, *key*, *value*⟩, where *table* is the name of the table (only FoodPrice or FoodQuantity), *key* is the common key of two tables, and *value* is the value at the corresponding record in FoodPrice or FoodQuantitty.

- The file contains digits, lowercase letters, white spaces, and newline characters.
- Two consecutive elements are separated by a whitespace. An item of type string is always a single word (no whitespaces in between).

Sample input:

Files	file01	file02
Content	FoodPrice Pizza 8 FoodPrice Burger 6 FoodQuantity Pizza 3	FoodPrice FrenchFries 4 FoodQuantity Burger 5 FoodQuantity Cake 2

Main requirement: Perform RIGHT OUTER JOIN FoodPrice table to FoodQuantity table.

- A value of null is used to represent the fact that there is no matched value.

Output: Tuples of (*key, first_value, second_value*), where *key* is the common key of two tables, *first_value* and *second_value* are the corresponding values in the first table and the second table, respectively.

Sample output:

```
Pizza 8 3
Burger 6 5
Cake null 2
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 2

Question item 11

Input: Consider a database of two tables, FoodPrice and FoodQuantity, whose records are stored together in a collection of multi-line text files. Each line in a file is a tuple of three elements, ⟨*table*, *key*, *value*⟩, where *table* is the name of the table (only FoodPrice or FoodQuantity), *key* is the common key of two tables, and *value* is the value at the corresponding record in FoodPrice or FoodQuantitty.

- The file contains digits, lowercase letters, white spaces, and newline characters.
- Two consecutive elements are separated by a whitespace. An item of type string is always a single word (no whitespaces in between).

Sample input:

Files	file01	file02
Content	FoodPrice Pizza 8 FoodPrice Burger 6 FoodQuantity Pizza 3	FoodPrice FrenchFries 4 FoodQuantity Burger 5 FoodQuantity Cake 2

Main requirement: Perform INNER JOIN FoodPrice table to FoodQuantity table.

- The data guarantees that there will be at least one record after performing inner join.

Output: Tuples of (*key*, *first_value, second_value*), where *key* is the common key of two tables, *first_value* and *second_value* are the corresponding values in the first table and the second table, respectively.

Sample output:

```
Pizza 8 3
Burger 6 5
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 2

Question item 12

Input: Consider a database of two tables, FoodPrice and FoodQuantity, whose records are stored together in a collection of multi-line text files. Each line in a file is a tuple of three elements, ⟨*table*, *key*, *value*⟩, where *table* is the name of the table (only FoodPrice or FoodQuantity), *key* is the common key of two tables, and *value* is the value at the corresponding record in FoodPrice or FoodQuantitty.

- The file contains digits, lowercase letters, white spaces, and newline characters.
- Two consecutive elements are separated by a whitespace. An item of type string is always a single word (no whitespaces in between).

Sample input:

Files	file01	file02
Content	FoodPrice Pizza 8 FoodPrice Burger 6 FoodQuantity Pizza 3	FoodPrice FrenchFries 4 FoodQuantity Burger 5 FoodQuantity Cake 2

Main requirement: Perform FULL OUTER JOIN FoodPrice table to FoodQuantity table.

- A value of null is used to represent the fact that there is no matched value.

Output: Tuples of (*key, first_value, second_value*), where *key* is the common key of two tables, *first_value* and *second_value* are the corresponding values in the first table and the second table, respectively.

Sample output:

```
Pizza 8 3
Burger 6 5
FrenchFries 4 null
Cake null 2
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.

Level 3

Question item 13

Input: Consider a transactional database, whose transactions are stored in a collection of multi-line text files. Each line in a file is a key-value tuple of ⟨*key,* **set** of [*item*]⟩, where *key* is the identity of the transaction and *item* is an item bought in that transaction.

- The file contains digits, lowercase letters, white spaces, tabs, and newline characters.
- The key part and value part are separated by a single tab.
- Two consecutive items are separated by a white space.
- A set allows has no duplicate.

Sample input:

Files	Content
trans01	t01 hotdogs buns ketchup t02 hotdogs buns
trans02	t03 hotdogs coke chips t04 chips coke
trans03	t05 chips ketchup t06 hotdogs coke chips

Main requirement: Find all the frequent itemsets by using the **Apriori algorithm** with the specified minimum support in decimal (e.g., 0.5).

- The user enters the minimum support value via an argument in the command line.

Output: Key-value tuples of ⟨*itemset*, *occurrence*⟩, where *itemset* is the frequent itemset of k items and *occurrence* is the corresponding support value.

- The key part and value part are separated by a single tab.

Sample output: with minimum support = 0.5

```
hotdogs      4
chips        4
coke         3
chips coke   3
```

Technical requirement:

- Hadoop input/output formats should *effectively match the data format.*
- The amount of data transferred from the map phase to the reduce phase *must be minimal.*
- The order of output tuples does not matter. It is unnecessary to format the resulting numbers.