Project – Factory Line

1 Introduction

In *Rumeli Hisarüstü*, a factory called *1-A Factory* produces essential products. Every product has their respective value. In the factory, individual units called holders are responsible for handling the products in the factory line. Each holder is coupled with the previous and the next holder to create a product line. You are tasked with the organization of this product line.

2 Details

2.1 FactoryImpl

- This class should implement the `Factory` interface. The overridden methods should work the way they are described in the `Javadoc`.
- This class has three mandatory fields. These fields should be correctly modified inside the overriden methods.

```
private Holder first
private Holder last
private Integer size
```

- You are free to add helper methods.

2.2 Project1

- The main method should be implemented here. This class is the entry point of your program.
- You are required to read from the input file and write to the output file. These file paths will be given in the program arguments.
- Each input command will be given in a single line. You have to read the input file and write to the output file line by line.

3 Input & Output

3.1 Input

- **Add First -** Adds a new product at the beginning of the factory line.

  AF $product_{id}$ $product_{value}$

- **Add Last -** Adds a new product to the end of the factory line.

  AL $product_{id}$ $product_{value}$

- **Add -** Adds a new product to the given `index` of the factory line. Prints `"Index out of bounds."` if the index is out of bounds.

  A index $product_{id}$ $product_{value}$

- **Remove First -** Removes the first product from the factory line and prints it. Prints `"Factory is empty."` if the there is no product in the factory.

  RF

- **Remove Last -** Removes the last product from the factory line and prints it. Prints `"Factory is empty."` if the there is no product in the factory.

  RL

- **Remove Index -** Removes the product in the given `index` and prints it. Prints `"Index out of bounds."` if the index is out of bounds.

  RI index

- **Remove Product -** Removes the first occurrence of the product with the given `value` and prints it. Prints `"Product not found."` if the product is not in the factory line.

  RP $product_{value}$

- **Find -** Prints the product with the given $product_{id}$. Prints `"Product not found."` if the product is not in the factory line.

  F $product_{id}$

- **Get -** Prints the product in the given `index`. Prints `"Index out of bounds."` if the index is out of bounds.

  G index

- **Update -** Updates the `value` of the product with the given $product_{id}$ to $product_{value}$. Prints `"Product not found."` if the product is not in the factory line.

  U $product_{id}$ $product_{value}$

- **Filter Duplicates -** Removes products such that after the filtering process there is only a single occurrence of each product in the factory line. In the context of this method, duplicate products are products with equal `value` fields, the `id` fields are of course unique.

  FD

- **Reverse -** Reverses the factory line.

  R

- **Print -** Prints the factory line.

  P

Input file path will be given as the first program argument.

3.2 Output

- **Product -** Products will be printed in the format below.

  (product_{id}, product_{value})

- **Factory Line -** Factory line will be printed in the format below.

  { product_1, product_2, ... , product_n }

Output file path will be given as the second program argument.

Input	**Corresponding Output Line**	**Explanation**
P	{}	Initially the factory line is empty.
RF	Factory is empty.	Cannot remove from the front since the factory is empty.
RL	Factory is empty.	Same as above, but now it's the end.
AF 3 9		$(3, 9)$ is added to the front.
A 0 2 4		$(2, 4)$ is inserted to `index` 0.
AL 5 25		$(5, 25)$ is added to the end.
A 0 1 1		$(1, 1)$ is inserted to `index` 0.
A 3 4 16		$(4, 16)$ is inserted to `index` 3.
AL 6 36		(6,36) is added to the end.
AF 7 30		$(7, 30)$ is added to the front.
P	{(7, 30),(1, 1),(2, 4),(3, 9), (4, 16),(5, 25),(6, 36)}	Factory line is printed.
F 3	(3, 9)	Product with `id` 3 is found and printed.
F 20	Product not found.	Product with `id` 20 does not exist.
U 4 9	(4, 16)	`value` of the product with `id` 4 is updated to 9. The previous product is printed.
U 13 21	Product not found.	Product with `id` 13 does not exist.
G 0	(7, 30)	Product with `index` 0 is printed.
G 17	Index out of bounds.	The factory line has 7 products so `index` 17 is out of bounds.
U 1 36	(1, 1)	`value` of the product with `id` 1 is updated to 36. The previous product is printed.
A 21 21 21	Index out of bounds.	There is no product at `index` 21.
FD	2	All duplicates are removed from the factory line. In this case it was $(4, 9)$ and $(6, 36)$. Prints the number of duplicates removed.
P	{(7, 30),(1, 36),(2, 4), (3, 9),(5, 25)}	Prints the factory line.
R	{(5, 25),(3, 9),(2, 4), (1, 36),(7, 30)}	Reverses the factory line. Prints the new version.
RP 9	(3, 9)	Removes the first product with `value` 9.
RP 13	Product not found.	No product has `value` 13.
RL	(7, 30)	Removes and prints the last product.
RI 2	(1, 36)	Removes and prints the product at `index` 2.
RF	(5, 25)	Removes and prints the first product.
RI 4	Index out of bounds.	The factory line has a single product so `index` 4 is out of bounds.

Table 1: Line by line analysis of example input