Java → Functional streams → Currying

Java → Reducer operator

Hard 22 minutes

56 users solved this problem. Latest completion was **4 days ago**.

Code Challenge — Write a program

Write three operators:

1. **A reduce operator** that accepts an initial value (seed) and a combiner function and then returns a new function that combines all values in the given integer range (inclusively) into one integer value (it's a simple form of reduction).
2. In terms of **the reduce operator** define a **sum operator** for summing integer values in the given range.
3. In terms of **the reduce operator** define a **product operator** for multiplying integer values in the given range.

Try not to use Stream API. Write the reducer yourself.

To simplify the problem all functions are declared, **you need to finish their realization**.
Look carefully at definition of each operator.

During testing all three operators will be tested. **The left boundary <= the right boundary**.

Example 1. Left boundary = 1, right boundary = 4.

- **sumOperator** returns the result 10.
- **productOperator** returns the result 24.

Example 2. Left boundary = 5, right boundary = 6.

- **sumOperator** returns the result 11.
- **productOperator** returns the result 30.

Code Editor IDE

Java

```
/**
 * The operator combines all values in the given range into one value
 * using combiner and initial value (seed)
 */
public static final BiFunction<Integer, IntBinaryOperator, IntBinaryOperator> reduceIntOperator =
        (seed, f) -> (left, right) -> IntStream.rangeClosed(left, right).reduce(seed, f);
/**
 * The operator calculates the sum in the given range (inclusively)
 */
public static final IntBinaryOperator sumOperator =
        (left, right) -> IntStream.rangeClosed(left, right).reduce(Integer::sum).orElse(0);

/**
 * The operator calculates the product in the given range (inclusively)
 */
public static final IntBinaryOperator productOperator =
        (left, right) -> IntStream.rangeClosed(left, right).reduce(1, (x, y) -> x * y);

```

✓ **Correct, but can be improved**

10 users liked this problem. **2** didn't like it. **What about you?**

Run Continue Solve again

Time limit: 8 seconds Memory limit: 256 MB

Comments (8) Hints (2) Useful links (0) Solutions (1)

Share something, Sergey Kubatko

Post Please do not post solutions here

Sort by: Last posted ▾

U1 **User 1074039** 3 months ago Report

Hopefully, my description will help somebody. We should use the seed as an initial value to perform combine operation with the left boundary value. After that, we should combine the result of the previous calculation with the next value in the given range. The next tutorial can give some hints as well - https://hyperskill.org/learn/step/3569.

♡ 0 Reply

JS **Jacek Sawko** 3 months ago Report

Can You please explain what do we need to do with first Function? Why do we need seed? How do we combine theses functions?

♡ 0 Reply

U1 **User 1074039** 3 months ago Report

I agree with Michał Borciuch. From the description, it's not obvious what should be done with the seed. Spent 2h with dome tests but still can not pass the first test (

♡ 0 Reply

MB **Michał Borciuch** 3 months ago Report

Why do we need the seed? What is his role? All my operators work fine, but i can't pass tests because of no.

♡ 2 Reply

> AG **Anton Gunkin** about 2 months ago Report
>
> Because you are working with binary operators, so if you are given a range with one element, in order to apply this operator you need one more element
>
> ♡ 0 Reply
>
> > AG **Anton Gunkin** about 2 months ago Report
> >
> > It's also used to apply the very first operation
> >
> > ♡ 0 Reply

C **christianp** 5 months ago Report

hint:
the primitive initial values should be supplied by you during the sumOperator and productOperator declarations.
0 is a good initial value for addition, but ruins things for multiplication.

♡ 0 Reply

EB **Evgenij Bulygin** 5 months ago Report

Please add right answer to test file

♡ 0 Reply