Relational MapReduce Patterns #2 - Selections

HackerRank

Mappers and Reducers

Here's a quick but comprehensive introduction to the idea of splitting tasks into a MapReduce model. The four important functions involved are:

```
Map (the mapper function)
EmitIntermediate(the intermediate key,value pairs emitted by the mapper functions)
Reduce (the reducer function)
Emit (the final output, after summarization from the Reduce functions)
```

We provide you with a single system, single thread version of a basic MapReduce implementation.

Task

Given a set of integers, only select and display odd integers, greater than 10.

The code for the MapReduce class, reading and splitting the text, parts related to IO etc. has already been provided. However, for this particlar task, you ONLY need parts of the mapper and its related functions. However certain parts of the mapper and reducer functions are incomplete. You need to replace the questionmarks (?). Your task is to fill up these question marks appropriately, such that the program works, and performs the specified task.
Also, this program may output certain information to the error stream. This information has been logged to help beginners gain a better understanding of the the intermediate steps in a map-reduce process.

Language Support

Java and Ruby.

Input Format

The first line contains an integer **N**, which is the number of elements in the set. This is followed by **N** integers, each on a new line, such that -100 <= **X** <= 100. Also, 10 <= **N** <= 100
For instance, if **R** = [10,20,40,20,60];you may treat it as [10,20,40,60]

Output Format

Output each of the integers satisfying the predicate (odd numbers exceeding 10), each on a new line, without disturbing their relative ordering.

Sample Input

```
10
99
43
-27
```

```
21
99
-100
35
-91
-45
59
```

Sample Output

```
99
43
21
99
35
59
```

Explanation

The first array has 10 elements. We finally output only the odd integers, exceeding 10, without changing their relative ordering.