Worksheet: Map/Reduce

You are given the following parallel Map/Reduce “framework” for processing a collection of Strings using Java Streams:

```
List<String> list = Arrays.asList("Rice", "Owls", "are", "the", "best");
var value =
    list.stream().parallel()
        .filter(_____A______)
        .map(_____B_____)
        .reduce(_____C_____);
```

Using this framework, solve the following problems by filling in the blanks A, B and C (note that C can be 1, 2 or 3 arguments, depending on which variant of reduce you choose):

1. Find all the strings that contain the letter “s”, convert them all to upper case, and concatenate them
2. Find all the strings of length 4, repeat each of them twice (i.e. “are” becomes “areare”), then find the smallest of them in lexicographical order
3. Find the total length of all the strings that start with a lowercase letter

Worksheet: Map/Reduce

1. Find all the strings that contain the letter “s”, convert them all to upper case, and concatenate them

```
List<String> list = Arrays.asList("Rice", "Owls", "are", "the", "best");
var value =
    list.stream().parallel()
        .filter(e -> e.contains("s"))
        .map(e -> e.toUpperCase())
        .reduce("", String::concat, String::concat);
```

“OWLSBEST”

Worksheet: Map/Reduce

2. Find all the strings of length 4, repeat each of them twice (i.e. “are” becomes “areare”), then find the smallest of them in lexicographical order

```
List<String> list = Arrays.asList("Rice", "Owls", "are", "the", "best");
var value =
    list.stream().parallel()
        .filter(e -> e.length() == 4)
        .map(e -> e.repeat(2))
        .reduce((a,b) -> a.compareToIgnoreCase(b) < 0 ? a : b);
```

```
Optional[“OwlsOwls”]
```

Worksheet: Map/Reduce

3. Find the total length of all the strings that start with a lowercase letter

```java
List<String> list = Arrays.asList("Rice", "Owls", "are", "the", "best");
var value =
    list.stream().parallel()
        .filter(e -> e.charAt(0) >= 'a' && e.charAt(0) <= 'z')
        .map(String::length)
        .reduce(Integer::sum);
```

```
Optional[10]
```