MapReduce of Palindromes

Find frequencies of palindromes for a set of given text documents. Assume that there is a defined function `is_palindrome(str)`, which returns `True` if a given `str` is a palindrome, otherwise, `False`.

Problem

Given a set of text documents, find frequencies of all palindromes.

Sample Input

```
today level ok dont civic
tomorrow level madam civic yes level
```

Mapper

```
// comment: assume that k is a record number of input file, ignored
// comment: v is the entire input record
map(k,v) {
    words=v.split(" ") #split words by space
    for (w in words) {
        if(is_palindrome(w)==True) {
            # w is a palindrome
            emit(w,1)
        }
    }
}
```

Output of Mappers

```
(level,1)
(civic,1)
(level,1)
(madam,1)
(civic,1)
(level,1)
```

Output of Sort and Shuffle

```
(level, [1, 1, 1])
(civic, [1, 1])
(madam, [1])
```

Reducer

```
// k is a unique palindrome
// values is an Iterable<Integer>
reduce(k,values) {
    count = 0
    for(v : values) {
        #sum of count of palindrome words
        count = count + v
    }
    # now, output the final count for a palindrome
    emit(k,count) #return word & its total count as output
}
```

Output of Reducers

```
(level,3)

(civic,2)

(madam,1)
```

Question-1: Write a combiner for this MapReduce

Question-2: How do you prove that your combiner is correct?