# Quiz 9: MapReduce

**Due** Jul 20 at 11:59pm **Points** 100 **Questions** 5
**Available** Jul 14 at 8am - Jul 20 at 11:59pm 7 days **Time Limit** None

## Instructions

For each question in this quiz, provide a pseudo-code solution. Assume an operation emit(k,v) for emitting a key-value pair in the output. Use syntax of the form

```
for each value in values {
   do something with value
}
```

for looping through a list of values.

This quiz was locked Jul 20 at 11:59pm.

## Attempt History

| | Attempt | Time | Score |
|---|---|---|---|
| **LATEST** | Attempt 1 | 52 minutes | 65 out of 100 |

Score for this quiz: **65** out of 100
Submitted Jul 20 at 4:47pm
This attempt took 52 minutes.

### Question 1

**5 / 20 pts**

Sort: Given as input a list of key-value pairs, output that list in sorted order. Allow for the same key to occur more than once in the input list, and output a key-value pair for each such occurrence.

```
map (k, v) { ... }
```

```
reduce (k, values) { ... }
```

Your Answer:

```
map (k, v) {
        for each value in v {
                emit(value, 1)
        }
}
reduce (k, values) {

        List <Entry<k, values>> list = new ArrayList<> (k.entrySet());
        list.sort(Entry.comparingByValue());
        return list
}
```

map (k, v) { return emit(k,v) } reduce (k, vs) { foreach v in vs { emit(k,v) } }

## Question 2

**20 / 20 pts**

Word Count: Given as input a list of text lines, output a list of the pairs (word,n) in the text, where n is the number of times that word occurs in the text.

```
map (offset, line) { ... }
```

```
reduce (word, values) { ... }
```

Your Answer:

```
map(offset, line) {
        String [] words = line.split(" ");
        for each word in words {
                emit(word, 1);
        }
}
reduce(word, values) {
        Integer n = 0;
        for each value in values {
                n = n + value
        }
        emit(word, n)
}
```

## Question 3

**20 / 20 pts**

More than a million hits: Given as input a list of text lines taken from Web server logs, where each line includes the URL for the Web page accessed, output a list of URLs for the pages on the Web server that have more than one million hits.

```
map (lineNum, line) { ... }
```

```
reduce (url, values) { ... }
```

Your Answer:

```
map (lineNum, line) {
        for each l in line {
                emit(l, 1)
        }
}
reduce (url, values) {
        Integer hits = 0;
        for each value in values {
                hits = hits + value
        }
        if hits >= 1000000 {
                emit(url, hits)
        }
}
```

## Question 4

**10 / 20 pts**

Links: Given as input a list of Web pages taken from a Web site, output a list of pairs (link, list [url1,…,urln]) where link is a link to one of the pages at the Web site, and url1,…,urln are the URLs for the other Web pages at the Web site that link to it.

```
map (url, content) { ... }
```

```
reduce (link, urls) { ... }
```

Your Answer:

```
map (url, content) {
        for each c in content {
                emit(c, 1)
        }
}

reduce (link, urls) {

        List [] list = new List

        for each url in urls {

                link.append(url)

        }

        emit (link, list)

}
```

map (url, line) { for each link l in line { emit(l, url) } }

## Question 5

**10 / 20 pts**

Most Popular Pages: Given as input a list of text lines taken from Web server logs, output a list of pairs (url,n) where n is the number of times the Web page with URL url is accessed, and with the list ordered by most popular Web pages. In this case, you will require two passes of map-reduce.

```
map1 (offset, line) { ... }
```

```
reduce1 (url, values) { ... }
```

```
map2 (url, n) { ... }
```

```
reduce2 (n, urls) { ... }
```

Your Answer:

```
map1 (offset, line) {

        for each l in line {
```

```
        emit(l, 1)
    }
}
reduce1 (url, values) {
    Integer n = 0;
    for each value in values {
        n = n + value
    }
    emit(url, n)
}
map2 (url, n) {


}
map2 (url, n) {
    for each n in n {
        emit(n, 1)
    }
}
reduce2 (n, urls) {
    List <Entry<n, url>> list = new ArrayList<> (urls.entrySet());
    list.sort(Entry.comparingByValue());
    return list
}
```

map2 (url, n) { emit(n, url) } reduce2 (n, urls) { for each url u in urls { emit(u,n) } }

Quiz Score: **65** out of 100