Ex 1.

a)

Map/Reduce pseudocode:

Map 1:

```
For each input (linenumber,text) pair {
    ip = extract_IP(text)); //gives users identified as IP address
    browser = extract_browser(text); //gives the browser used for that log
    Output key-value pair (ip, browser);
}
```

Shuffle 1:

Group all key/value pair with same key into one key/value-list pair:
(ip, (browser1,browser2…, browser n));

Reduce 1:

For all key/value-list pair, output identify.
Output (ip, (browser1,browser2…, browser n));

Map 2(using output of MapReduce 1 as input):

```
For each input (ip, (browser1,browser2…, browser n){
    For each browser of value-list (browser1, browser2,..., browser n){
        Output key-value pair (browser, 1);
        //i.e (browser1,1); (browser2, 1); … (browser n, 1)
    }
}
```

Shuffle 2:

Same concept as shuffle 1, with the output key-value pair:
(browser, (1,1,1…))

Reduce 2:

```
For each key/value-list pair (browser, (c1,c2,...c n)){
    If(n > 100){
        Output (browser, n);
    }
}
```

a)

Mapper 1:

(going to put only one log as example as it is pretty long)

Input:

(0, 169.122.23.15 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache pb.gif HTTP/1.0" 200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)");

Output:

(169.122.23.15, Mozilla/4.08)

Reducer 1 *(assume shuffled as described above):*

Input: (169.122.23.15, (Mozilla/4.08, Chrome/1.0))

Output: (169.122.23.15, (Mozilla/4.08, Chrome/1.0))

Mapper 2:

Input: (169.122.23.15, (Mozilla/4.08, Chrome/1.0))

Output: (Mozilla/4.08,1) ; (Chrome/1.0, 1)

Reducer 2 *(assume shuffled, assume processed not only this one log, but all logs)*:

Input: (Mozilla/4.08,(1,1,1,1…)); (Chrome/1.0, (1,1,1,1…)), … (*and other browsers)*

Output: (Mozilla/4.08, 500); (Chrome/1.0, 505); (Mozilla/4.2, 360); *//last example is to show we consider different versions of same browsers, different browsers.*

b) It would reduce the amount of I/O. For example, in our implementation above, a useful combine would be to combine the output data of Mapper 2 (i.e. (Mozilla/4.08,1);(Mozilla/4.08,1) -> (Mozilla/4.08,2)). Doing so, it would reduce data to be transferred to shuffle, but the amount of data received by reducer should still be the same after the shuffling process. Nevertheless, the combine would be useful as it will still reduce the data processed, thus the amount of I/O.

Ex 2.

a)

//MapReduce 1 would be the same as Ex.1, until the Reduce part

Map 1:

```
For each input (linenumber,text) pair {
        ip = extract_IP(text)); //gives users identified as IP address
        browser = extract_browser(text); //gives the browser used for that
log
        Output key-value pair (ip, browser);
}
```

Shuffle 1:

Group all key/value pair with same key into one key/value-list pair:
(ip, (browser1,browser2…, browser n));

Reduce 1:

```
For all key/value-list pair (ip, (browser1,browser2…, browser n))
{
        if(value-list.size() > 1){ //get all users who used more than one
browser
                Output (ip, (browser1,browser2…, browser n));
        }
}
```

Map 2(using output of MapReduce 1 as input):

```
//we now have all users who used more than one browse
        For each input (ip, (browser1,browser2…, browser n){
                Output (ip,1);
        }
```

Shuffle 2:

//because the question only wants the total number, we can initialize a counter as output:

```
Int userNum = 0;
```

Because all Ip should be unique after MapReduce 1, no Shuffle needed,
Only need to increment counter:

```
For each output (ip,1){ userNum++;}
```

Reduce 2:

```
Output userNum;
```

b) For my implementation, the last reducer step of my workflow only needed ONE reducer process that will output the counter userNum.

Q6.

```
grunt> describe joined
joined: {
    popvax::highpop::country: chararray,
    popvax::highpop::population: int,
    popvax::vdata::country: chararray,
    popvax::vdata::iso3: chararray,
    popvax::vdata::who_region: chararray,
    popvax::vdata::persons_fully_vaccinated: int,
    vmeta::iso3: chararray,
    vmeta::vaccine_name: chararray,
    vmeta::product_name: chararray,
    vmeta::company_name: chararray
}
```