**Week 3 assessment**

**Web Server Logs**

10 questions

1.

Suppose a web log is modified to now have a sixth piece of information, a priority, that can be represented as a String.

Which one of the following is the least likely change to the LogEntry class to accommodate this new part of a web log?



The **toString** method is modified to include a String parameter. X



The **toString** method is modified to include the new priority as part of the return String.



A new **getPriority** method is created to return the priority.



A new String parameter is added to the constructor.



A new private variable named **priority** is created.



A new String field is initialized in the constructor.

2.

Consider the following code for the **readFile** method of the **LogAnalyzer** class.

public void readFile(String filename) { FileResource fr = new FileResource(filename); for (String line : fr.lines()) { LogEntry le = WebLogParser.parseEntry(line); } }

In the **Tester** class, **readFile** is called with a correct filename, and then **printAll** is called, but nothing is printed.

Which one of the following is likely the best reason why?



In **readFile**, the log entries were not stored in **records**. X



In **readFile**, a **System.out.println** statement is missing from the body of the for loop.



In **readFile**, a **System.out.println** statement is missing that should be right before the for loop.



In **readFile**, a **System.out.println** statement is missing that should be right after the for loop.



In **readFile**, the wrong parameter is sent to **parseEntry**.

3.

Consider the following code for the method **printAllHigherThanNum** with one integer parameter **num**. This method should print all the logs that have a status code higher than **num**.

Which one of the following would be the best choice for suitable code for this method?



for (LogEntry le : records) { if (le.getStatusCode() > num) { System.out.println(le); } else { System.out.println(); } }



if (le.getStatusCode() > num) { for (LogEntry le : records) { System.out.println(le); } } else { System.out.println(); }



X for (LogEntry le : records) { if (le.getStatusCode() > num) { System.out.println(le); } }



if (le.getStatusCode() > num) { for (LogEntry le : records) { if (le.getStatusCode() > num) { System.out.println(le); } } }



if (le.getStatusCode() > num) { for (LogEntry le : records) { System.out.println(le); } }

4.

Run the method **countUniqueIPs** on the file **weblog2\_log**.

How many unique IP addresses are in the file? 45



5.

Run the method **uniqueIPVisitsOnDay(“Sep 24”)** on the file **weblog2\_log**.

What size is the ArrayList that is returned? 14



6.

Run the method **countUniqueIPsInRange(200,299)** on the file **weblog2\_log**.

What number is returned? 40



7.

Run the method **mostNumberVisitsByIP** after a HashMap has been created from the method **countVisitsPerIP** on the file **weblog2\_log**.

What number is returned? 63



8.

Run the method **iPsMostVisits** after a HashMap has been created from the method **countVisitsPerIP** on the file **weblog2\_log**.

What single IP address is returned in the ArrayList?



103.57.41.178



188.162.84.63 x



200.69.213.251



210.4.104.99



212.128.74.248

9.

Run the method **dayWithMostIPVisits** with a HashMap has been created from the method **iPsForDays** on the file **weblog2\_log**.

What day is returned?



Sep 24 x



Sep 26



Sep 28



Sep 30

10.

Run the method **iPsWithMostVisitsOnDay** with two parameters—one, a HashMap that has been created from the method **countVisitsPerIP** on the file **weblog2\_log** and two, the day “Sep 29”.

One IP address is returned in the ArrayList—what is it?



103.57.41.178



188.162.84.63



210.4.104.99



212.128.74.248 x



212.185.210.111