**Basic Analysis** (70 points)
Please answer any ten of the following questions (Q1 to Q13).

```
The total query documents are: 2477283
```

Q1. What is the average number of queries per user id?

```
537639 unique userid observed
2477283 query processed
Average number of queries per user id is 4.607.
```

Q3. What percentage of queries are mixed case? All upper case? All lower case?

```
All lower case: 1660630
All upper case: 108582
Mixed case: 708071
Percentage of queries are mixed case: 28.58%
Percentage of queries are all upper case: 4.38%
Percentage of queries are all lower case: 67.03%
```

Q4. What percent of the time does a user request only the top 10 results?

```
Top 10 only: 1920997
2477283 query processed
Top 10 percent= 77.54%
```

Q6. What are the 20-most common queries issued?

```
0    sex, numOccurs(4660)
1    yahoo, numOccurs(3129)
2    internal site admin check from kho, numOccurs(2191)
3    chat, numOccurs(1520)
4    porn, numOccurs(1498)
5    horoscopes, numOccurs(1315)
6    pokemon, numOccurs(1284)
7    SiteScope test, numOccurs(1283)
8    hotmail, numOccurs(1223)
9    games, numOccurs(1163)
10   mp3, numOccurs(1151)
11   weather, numOccurs(1140)
12   www.yahoo.com, numOccurs(1127)
13   maps, numOccurs(1110)
14   yahoo.com, numOccurs(1036)
15   ebay, numOccurs(983)
```

```
16    recipes, numOccurs(980)
17    britney spears, numOccurs(888)
18    horoscope, numOccurs(868)
19    xxx, numOccurs(865)
20    jokes, numOccurs(832)
```

I am surprised pokemon/britney spears are in the hot search list, but it makes more sense I realized the query log is from 1999.

Q7. What are the 20 most common non-stopwords appearing in queries?

```
0     find, numOccurs(147234)
1     com, numOccurs(94147)
2     www, numOccurs(60170)
3     free, numOccurs(57778)
4     pictures, numOccurs(35862)
5     sex, numOccurs(32576)
6     information, numOccurs(31120)
7     christmas, numOccurs(28326)
8     nude, numOccurs(27157)
9     pics, numOccurs(18895)
10    buy, numOccurs(17356)
11    online, numOccurs(15922)
12    web, numOccurs(14474)
13    music, numOccurs(14037)
14    women, numOccurs(13823)
15    games, numOccurs(13585)
16    cards, numOccurs(11543)
17    porn, numOccurs(11314)
18    stories, numOccurs(10976)
19    site, numOccurs(10692)
20    codes, numOccurs(9789)
(no stemming)
```

Q8. What percent of queries contain stopwords like ‘and’, ‘the’, ‘of’, ‘in’, ‘at’?

```
Percent of queries: 26.42%
Query contains stopwords: 654542
2477283 query processed
```

Q9. What are the 10 most common non-stopwords appearing in queries that contain the word *download*?

```
0     free, numOccurs(1374)
1     games, numOccurs(600)
2     mp3, numOccurs(460)
3     game, numOccurs(379)
4     music, numOccurs(304)
5     downloadable, numOccurs(260)
6     +download, numOccurs(194)
7     pokemon, numOccurs(182)
8     full, numOccurs(157)
9     software, numOccurs(147)
10    mac, numOccurs(127)
11    com, numOccurs(126)
12    movies, numOccurs(120)
13    funny, numOccurs(112)
14    video, numOccurs(100)
15    christmas, numOccurs(97)
```

Q10. What percentage of queries were asked by only one user?

```
Only one user query: 223351
Percentage: 9.02%

I can tell most user should query more than once.
```

Q12. Which occurs in queries more often "Al Gore" or “Johns Hopkins"? "Johns Hopkins" or "John Hopkins"?

```
Al Gore: 34
Johns Hopkins: 33
John Hopkins: 9

“Al Gore occurs” more often than “Johns Hopkins”
“Johns Hopkins” occurs more often than “John Hopkins”
```

Q13. How often do URLs appear in queries?

```
URL: 100529
Percentage of URL appear in queries:4.05%
```

**Other Analysis** (30 points)
Answer any three of the following questions (Q14 to Q20).

Q15. Can you find addresses, phone numbers, and other identifiers in the log file? Is it likely that this web query log puts anyone's privacy at risk? Justify your response.

I started try to use state abbreviation to find address, but it turns out too many results. Interesting finding is the the rank of state query, top3 are CA, NY, and PA:

```
0     CA, numOccurs(1335)
1     NY, numOccurs(1091)
2     PA, numOccurs(860)
3     NJ, numOccurs(729)
4     NC, numOccurs(550)
5     MA, numOccurs(499)
6     VA, numOccurs(483)
7     TX, numOccurs(420)
8     FL, numOccurs(408)
9     IL, numOccurs(389)
10    CO, numOccurs(368)
```

Then I use “Rd.” and “Ave.” to search query, then it did return address such as:

```
“can you tell me the phone listing for the address at 1508 Monterey Rd., Florence, MS 39073”
“What houses are on 116th St. and Towen Rd. in Indiana”
“8750 Mc Kinney Rd.”
“Sloppy Joes at 31 Fortune Rd.”
“318 East Garden Rd. Pittsburgh PA  15227”
“who live at 729 evergreen Rd. Severn MD 21144”
“13008 Osage Rd. Louisville Kentucky”
“Driving directions 9321 MemorialPark Rd. West Palm Beach Fl.”
“What is the zip code on Mather Field Rd. in Sacramento, CA”
“who lives at 2411-4, Selwyn Ave., Charlotte, NC, 28209
“Brandon Siljord, 400 Pine Ave., Little Falls, Mn, 56345  , 320-632-8172
“DSI Automotive products 1650 E. Main Ave. Box 699 West Fargo,ND 58078-0699
“Days Inn/ 11133 Washington Ave. St. Louis Mo.
“what are the name of the apartments at 5000 Ave. K in Plano, Texas.
“Coastal Neurological Medical Group, Inc  9850 Genesee Ave.  La Jolla, CA
“who lives in 108 Sitton Ave. in Gordon,GA?
“inexpensive cafe/restaurant near 2101 Constitution Ave., NW, Washington, DC 20418”
```

“where can i find information on Queen of Peace High School, 7659 S. Linder Ave., Burbank, IL, 60459-1393”
“Village Green Apts. 21230 Homestead Ave.”

When to find phone numbers, it does returns some phone number such as:

908-534-1415
1223456789
0883622076
0883949725
WHERE IS 225-704-1418?
4001049801
who lives at 518-229-5643
who belongs to the telephone number 518-229-5643
0884412784
Whose phone number is this 518-842-6004?
6174697914
5325301078
201-261-1880
781-322-9200
where in the US is phone number 416-777-8781
732-223-5300 mallard park
"301-330-0333"
617-323-2082
800-555-1212
3436533048
800-245-2133
whose telephone number is 732-828-7370
What is the name of the company with the phone number 978-851-2000?
What is the address for phone number 510-235-2864
I would like to know who has this telephone number:  402-471-5540 ?
holiday inn jamacia 876-953-2486
Brandon Siljord, 400 Pine Ave., Little Falls, Mn, 56345 , 320-632-8172
0961376600
What property management company in south orange county has the fax Brandon Siljord, 400 Pine Ave., Little Falls, Mn, 56345 , 320-632-8172
Who's telephone number is 316-371-6331
Who's telephone number is 316-267-0876
what business is the number 212-725-4300
what listing is the following 212-725-4300
Who is  416-977-4605
Who uses this number  416-977-4605
whos phone number is 815-645-8504?
whose phone number is 817-297-9754
who's telephone number is this 856-401-1882
who's telephone number is this 856-401-1882

What business does 941-421-2516 belong to?
Where can I look up this phone number 941-421-2516?
who's phone number is 856-303-0819
who belongs to this phone number 425-670-6228
who's phone number is 856-303-0819
who last had this phone number 516-773-3682
find resident of this phone number 253-520-5080
Whose telephone number is 208-323-4663?

However except for phone number, most user query for missed called number.

Later on I try to use “credit card number” to see if can find any credential information, it turns out some interesting result:

“credit card number generator”
“credit card number generator download”
“Fake credit card number”
“Where can i find credit card numbers on-line?”
“Where can i find fake credit card numbers on-line?”
“how to generate credit card numbers?”
“fake credit card numbers”
“Where can I find free pictures of Britney Spears naked without giving”

In general, I would think most users don't put personal information in query but in some cases might be someone they are interested in. Most of case users query for famous or government, school, hotel type information.

## Q18. How does query volume change throughout the day?

09:00~10:00 346644 queries
10:00~11:00 337621 queries
11:00~12:00 335016 queries
12:00~13:00 328596 queries
13:00~14:00 323279 queries
14:00~15:00 288767 queries
15:00~16:00 262841 queries
16:00~17:00 254519 queries

I am surprised the result, since I guess it might be more query volume about noon or late afternoon. The result shows the peak query volume is from 09:00~10:00, then gradually decrease. The least query volume is from 16:00~17:00

Q19. What are the most popular websites mentioned in the queries?

```
0     www.yahoo.com, numOccurs(1127)
1     yahoo.com, numOccurs(1036)
2     hotmail.com, numOccurs(661)
3     www.hotmail.com, numOccurs(616)
4     amazon.com, numOccurs(612)
5     aol.com, numOccurs(414)
6     www.pokemon.com, numOccurs(197)
7     gay.com, tnumOccurs(181)
8     weather.com, numOccurs(179)
9     www.aol.com, numOccurs(178)
10    www.bluemountain.com, numOccurs(169)
11    iwon.com, numOccurs(168)
12    www.nick.com, numOccurs(166)
13    ebay.com, numOccurs(161)
14    nfl.com, numOccurs(154)
15    excite.com, numOccurs(153)
16    www.excite.com, numOccurs(145)
17    www.wwf.com, numOccurs(141)
18    www.pch.com, numOccurs(140)
19    www.ampland.com, numOccurs(137)
```

I realized the query log is from 1999, which Google is still startup search engine. Yahoo is the most popular website at that time, and hotmail is the next popular website.

**Extra Credit** (a few meager points)
You might come up with other interesting questions. Very interesting analysis might get a couple of measly extra-credit points.

As most search engines try to use query log as business decision, and use advertising as main income, I searched several categories such as “restaurant”, “hotel”, “shopping” and try to retrieve user query interests back to 1999:

“restaurant: ”

```
valentino restaurant, numOccurs(70)
evanston restaurants thai, numOccurs(30)
new england restaurant company, numOccurs(36)
san francisco restaurants, numOccurs(27)
restaurants of Aitkin, Minnesota, numOccurs(27)
```

“hotel:”,

where can i find hotels in venezuela, numOccurs(162)
las vegas hotels, numOccurs(102)
new york hotels, numOccurs(34)
chicago hotels, numOccurs(34)
hotels and motels in california, numOccurs(45)
ritz hotel barcelona, numOccurs(39)
1daytona beach hotels, numOccurs(36)
westin hotels, numOccurs(24)
hyatt hotels, numOccurs(24)

“shopping:”

online shopping, numOccurs(108)
saving food shopping money grocery households supermarket, numOccurs(287)
toronto shopping channel, numOccurs(96)
Mercata  shopping  sale -com, numOccurs(120)
comparison shopping, term_id(7), tf(2), df(30) and numOccurs(60)
adult gift shopping edibles, numOccurs(100)
sporting goods shopping, numOccurs(54)
shopping cart, numOccurs(36)
on line shopping malls, numOccurs(51)
toy shopping, numOccurs(34)
wedding shopping bridal shops, numOccurs(60)

**Deliverables**

Briefly describe the methods and tools that you used, and summarize any conclusions you reach.

I coded a simple Java program to analyze the query log.

**Source Code:**

```
// ---------------------------------------------------------------
// Chin-Ting Ko
//
// 605.744 Information Retrieval - Spring 2015 PA5
// ---------------------------------------------------------------
```

```
import java.io.*;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Scanner {
    String filename;
    BufferedReader reader = null;
    Dict dict1 = new Dict();
    Dict dict2 = new Dict();
    Dict dict3 = new Dict();
    Dict dict4 = new Dict();
    Dict dictQ = new Dict();

    public Scanner(String fname) throws IOException {
        filename = fname;
        reader = new BufferedReader(new InputStreamReader(new FileInputStream(filename)), 65536);
    }

    public void terminate() throws IOException {
        reader.close();
    }

    // Process input file line by line
    public void process() throws IOException {
        String line = null;
        while (true) {
            line = reader.readLine();
            //System.out.println(line);
            if (line == null) {           // Have reached EOF
                break;
            } else {
                processLine(line);
            }
        }
    }

    int docNum = -1;
    StringBuffer buf = null;

    public void processLine(String line) {
        if (line.length() < 1) { return; }
            if (line.length() >1) {      // start of doc
                 buf = new StringBuffer();
                 buf.append(line);
                 processTextForDoc();
            } else {
               return;
                //buf.append("\n");
            }
        }

  int numDocs = 0;
  int top10 = 0;
  int allLow=0;
  int allUpper=0;
  int mixCase=0;
  int stopword=0;
  int alGore=0;
  int johnsHopkins=0;
  int johnHopkins=0;
  int URL=0;
  int nine=0;
  int ten=0;
```

```
  int eleven=0;
  int twelve=0;
  int thirteen=0;
  int fourteen=0;
  int fifteen=0;
  int sixteen=0;

  // Very short stopword list
  static HashSet stopwords = new HashSet();
  static {
      String [] swords = {"OR", "IN", "which", "many", "when", "get", "my", "does", "having", "who", "where",
"how", "what", "the", "of", "to", "and", "a", "in", "for", "is", "that", "on", "with", "said", "by", "it",
"as", "be", "are", "at", "from", "not", "was", "or", "an", "this", "will", "he", "have", "has", "but", "its",
"new", "were", "they", "would", "more", "can", "we", "his", "other", "no", "been", "about", "also", "their",
"you", "than", "all", "up", "if", "may", "had", "there", "such", "after", "some", "into", "any", "out", "do",
"under", "use", "only", "these", "last", "am", "over", "so", "because", "most", "could", "each", "used",
"says", "support", "through", "between", "high", "did"};
      // which, many, where, how, who, when, what
      for(int i=0; i<swords.length; i++) {
          stopwords.add(swords[i]);
      }
        }

  static HashSet address = new HashSet();
  static {
      String [] state = {"AL", "AK", "NE", "AZ", "AR", "NH", "CA", "NJ", "CO", "NM", "CT", "NY", "DE", "NC",
"FL", "ND", "GA", "OH", "HI", "OK", "ID", "OR", "IL", "PA", "IN", "RI", "IA", "SC", "KS", "SD", "KY", "TN", "LA
", "TX", "ME", "UT", "MD", "VT", "MA", "VA", "MI", "WA", "MN", "WV", "MS", "WI", "MO", "WY"};
      // which, many, where, how, who, when, what
      for(int i=0; i<state.length; i++) {
          address.add(state[i]);
      }
        }

    public void processTextForDoc() {
        if (buf != null) {
            numDocs++;
            String text = buf.toString();
        //System.out.println("working on query#: "+numDocs);

            String column1= text.substring(0,text.indexOf(' '));
            //System.out.println("Col1 (Timestamp): "+column1);
            int column1Int=Integer.parseInt(column1);
            String column2to9= text.substring(text.indexOf('         ')+1);
            //System.out.println(column2to9);
            String column2= column2to9.substring(0,column2to9.indexOf('       '));
            //System.out.println("Col2 (Userid): "+column2);
            String column4= text.substring(text.lastIndexOf('        ')+1);
            String column1to3= text.substring(0,text.lastIndexOf('   '));
            String column3= column1to3.substring(column1to3.lastIndexOf('     ')+1);

            if (column3.equals("0")){
                 top10++;
            };
            //System.out.println("Col3 (FirstRank): "+column3);
            //System.out.println("Col4 (Query): "+column4);
            //System.out.println();

            StringTokenizer tok1 = new StringTokenizer(column1, "/.?!,;()\"' \t\n");
            Document doc1 = new Document();
            while (tok1.hasMoreTokens()) {
                String token = tok1.nextToken().toLowerCase();
                StringBuffer tokenStem = new StringBuffer(token);
                String tokenStemmed = tokenStem.toString();
```

```
    /*
    if (tokenStem.length()>4){
    tokenStemmed = tokenStem.delete(5, tokenStem.length()).toString();
    }
    */
    if (tokenStemmed.length() > 1 &&! stopwords.contains(tokenStemmed)) {// remove stop words

    //if (token.length() > 1) {
        doc1.addTerm(tokenStemmed);
        //System.out.println(tokenStemmed);
    }
}
dict1.handleDoc(docNum, doc1); // Update counts for words in this document

if(column1Int>=90000 &&column1Int<100000){
     nine++;
}
if(column1Int>=100000 &&column1Int<110000){
     ten++;
}
if(column1Int>=110000 &&column1Int<120000){
     eleven++;
}
if(column1Int>=120000 &&column1Int<130000){
     twelve++;
}
if(column1Int>=130000 &&column1Int<140000){
     thirteen++;
}
if(column1Int>=140000 &&column1Int<150000){
     fourteen++;
}
if(column1Int>=150000 &&column1Int<160000){
     fifteen++;
}
if(column1Int>=160000 &&column1Int<170000){
     sixteen++;
}

StringTokenizer tok2 = new StringTokenizer(column2, "/.?!,;()\"' \t\n");
Document doc2 = new Document();
while (tok2.hasMoreTokens()) {
    String token = tok2.nextToken().toLowerCase();
    StringBuffer tokenStem = new StringBuffer(token);
    String tokenStemmed = tokenStem.toString();
    /*
    if (tokenStem.length()>4){
    tokenStemmed = tokenStem.delete(5, tokenStem.length()).toString();
    }
    */
    if (tokenStemmed.length() > 1 &&! stopwords.contains(tokenStemmed)) {// remove stop words

    //if (token.length() > 1) {
        doc2.addTerm(tokenStemmed);
        //System.out.println(tokenStemmed);
    }
}
dict2.handleDoc(docNum, doc2); // Update counts for words in this document

/*
String column4Low=column4.toLowerCase();
String column4Upper=column4.toUpperCase();

if (column4Low.equals(column4)){
```

```
                allLow++;
            }
            else if (column4Upper.equals(column4)){
                allUpper++;
            }
            else{
                mixCase++;
            };
            if (column4.contains("Al Gore")){
                alGore++;
            }
            if (column4.contains("Johns Hopkins")){
                johnsHopkins++;
            }
            if (column4.contains("John Hopkins")){
                johnHopkins++;
            }
            */
            /*
            Document docQ = new Document();
            if (column4!=null){
            docQ.addTerm(column4);
            }
            /*
            if (column4.contains("http://")||column4.contains("www.")||column4.contains(".gov")||
column4.contains(".com")||column4.contains(".edu")){
                URL++;
                //System.out.println(column4);
               docQ.addTerm(column4);
            }*/

            //dictQ.handleDoc(docNum, docQ);



            StringTokenizer tok4 = new StringTokenizer(column4, "/.?!,;()\"' \t\n");
            Document doc4 = new Document();
            while (tok4.hasMoreTokens()) {
                 String tok=tok4.nextToken();
                //String token = tok4.nextToken().toLowerCase();
                //StringBuffer tokenStem = new StringBuffer(token);
                //String tokenStemmed = tokenStem.toString();
                /*
                if (tokenStem.length()>4){
                tokenStemmed = tokenStem.delete(5, tokenStem.length()).toString();
                }
                */

                String tel = "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$";
                Pattern pattern = Pattern.compile(tel);
                Matcher matcher = pattern.matcher(tok);

                if (tok.length() > 1 &&! stopwords.contains(tok) && column4.contains("shopping")) {// remove
stop words
                 //&& column4.contains("download")
                 //if (token.length() > 1) {
                    doc4.addTerm(column4);
                    System.out.println(column4);
                }
                /*
                if (tokenStemmed.length() > 1 && stopwords.contains(tokenStemmed)) {// remove stop words

                 stopword++;
                    //System.out.println(tokenStemmed);
```

```
                    break;
                }
                */
            }
            dict4.handleDoc(docNum, doc4); // Update counts for words in this document
        }
    }



    public static void main (String [] args) throws IOException {
        Scanner scanner = new Scanner(args[0]);
        PrintWriter output = new PrintWriter (new FileWriter("output.txt"));
        scanner.process();
        scanner.terminate();
        System.out.println(scanner.numDocs + " querys processed");
        /*
        System.out.println("Q19. What are the most popular websites mentioned in the queries? : ");
        scanner.dictQ.printStats(0, 20);
        System.out.println(scanner.numDocs + " querys processed");
        System.out.println(scanner.dict2.size() + " unique userid observed");
        /*
        scanner.dict1.printStats(0, 30);
        scanner.dict1.printStats(99, 100);
        scanner.dict1.printStats(999, 1000);
        scanner.dict1.printStats(9999, 10000);
        scanner.dict1.printStats(99999, 100000);
        scanner.dict1.printStats(999999, 1000000);

        System.out.println(scanner.numDocs + " querys processed");
        System.out.println(scanner.dict2.size() + " unique userid observed");
        scanner.dict2.printStats(0, 30);
        scanner.dict2.printStats(99, 100);
        scanner.dict2.printStats(499, 500);
        scanner.dict2.printStats(999, 1000);
        scanner.dict2.printStats(0, 2477283);
        System.out.println("Q10. What percentage of queries were asked by only one user? : ");

        /*
        double averageQuery = (double)scanner.numDocs/scanner.dict2.size();
        double Q4 = (double)scanner.top10/scanner.numDocs;
        double Q3 = (double)scanner.mixCase/scanner.numDocs;
        double Q8 = (double)scanner.stopword/scanner.numDocs;

        System.out.println();
        System.out.println("Q1. What is the average number of queries per user id? : "+ averageQuery);
        System.out.println("Top10 only: "+scanner.top10);
        System.out.println("Q3. What percentage of queries are mixed case? All upper case? All lower case? :
"+Q3);
        System.out.println("All lower case: "+ scanner.allLow);
        System.out.println("All upper case: "+ scanner.allUpper);
        System.out.println("Mixed case: "+ scanner.mixCase);
        System.out.println();
        System.out.println("Q4. What percent of the time does a user request only the top 10 results? : "+Q4);
        System.out.println();

        System.out.println("Q6. What are the 20-most common queries issued? : ");
        scanner.dictQ.printStats(0, 30);
        System.out.println();
        /*
        System.out.println("Q8. What percent of queries contain stopwords like ‘and’, ‘the’, ‘of’, ‘in’, ‘at’?:
"+Q8);
        System.out.println("Query contains stopwords: "+scanner.stopword);
```

```
        System.out.println();
        System.out.println("Q12. Which occurs in queries more often Al Gore or Johns Hopkins? Johns Hopkins or
John Hopkins?: ");
        System.out.println("Al Gore: "+scanner.alGore);
        System.out.println("Johns Hopkins: "+scanner.johnsHopkins);
        System.out.println("John Hopkins: "+scanner.johnHopkins);
        System.out.println();

        System.out.println("Q13. How often do URLs appear in queries? : ");
        System.out.println("URL: "+scanner.URL);

        System.out.println("Q18. How does query volume change throughout the day? : ");
        System.out.println("09:00~10:00 "+scanner.nine);
        System.out.println("10:00~11:00 "+scanner.ten);
        System.out.println("11:00~12:00 "+scanner.eleven);
        System.out.println("12:00~13:00 "+scanner.twelve);
        System.out.println("13:00~14:00 "+scanner.thirteen);
        System.out.println("14:00~15:00 "+scanner.fourteen);
        System.out.println("15:00~16:00 "+scanner.fifteen);
        System.out.println("16:00~17:00 "+scanner.sixteen);
        /*
        System.out.println(scanner.numDocs + " querys processed");

        */
        System.out.println(scanner.dict4.size() + " unique terms observed");
        System.out.println(scanner.dict4.numCounts() + " total terms encountered");
        System.out.println();
        scanner.dict4.printStats(0, 100);
        scanner.dict4.printStats(99, 100);
        scanner.dict4.printStats(499, 500);
        scanner.dict4.printStats(999, 1000);
        scanner.dict4.printStats(9999, 10000);

        System.out.println();
        }
}
```