CS 351 – DATA ORGANIZATION AND MANAGEMENT

FALL 2010 - Programming Project #3

Basic DNS Server Implementation with Linear Hashing

Due: January 3, 2010 at 11:59 PM

1. The Objective

In this programming assignment, you are asked to implement a JAVA console application, which simulates a basic DNS server using linear hashing file structure. The aim of the assignment is to make you understand linear hashing better and have fun with it. You are allowed to use your program for the second project. In fact, we encourage you to do so. Additionally, you will provide a short report (max. 1 page) that explains your algorithm to solve the problem. Also discuss whether linear hashing is a good method to store records of a DNS server. Justify your answer. **Please read the entire assignment before starting implementation.**

2. Problem Specification

The basic DNS server, which your program will simulate, provides IP addresses for domain names. In order to do this, it stores each (domain name, IP, expiration date) records until they expire. Your program will keep them in a linear hashing file structure. At each unit of time, some new records are inserted and some old records are deleted because they would not be valid anymore. First delete the records that are expired, then, add new records to the structure. Your program should implement these operations as well as answer coming queries (i.e., find the IP address of a given URL). In order to insert a record to the linear hashing structure, first find its hash value by using the provided hash function (see below). Then, insert the record into the structure by considering the hash value. Each query is a string representing a domain name. The input and output specifications are provided below.

Important Note: Input and output files are somewhat difficult to comprehend in this text since their lines are too long. Thus, please take a look at their actual versions on the web page.

Input Specifications

Your program will take five parameters.

- **Load Factor:** The load factor value of the linear hashing file structure. It is given as the <u>first parameter</u> to your program. (Note that insertions are done as we explained in the previous project and as we done in the classroom.)

  Sample Load Factor: 0.666

- **Number of Records per Bucket:** It is given as the <u>second parameter</u> to your program.

  Sample Number of Records per Bucket: 3

- **Records and Queries File:** In this file, each line *i* contains the records to be inserted and the queries at time *i*. The records are separated with a space character. The queries are also separated with a space character. Between records and queries, there exists a tab character. Each record contains three values (domain name, IP, expiration time). Each query is a domain name. The records and queries file path is given as the <u>third parameter</u> to your program.

  Sample Records and Queries File

  ```
  www.youtube.com 208.65.153.238 3 www.facebook.com 69.63.184.142 3
  www.wikileaks.ch 213.251.145.96 5 www.google.com 216.239.51.99 3
      www.youtube.com
  www.milliyet.com.tr 83.66.140.16 5 www.radikal.com.tr 213.243.16.14 5
      www.facebook.com
  www.xkcd.com 72.26.203.99 8  www.yahoo.com 69.147.76.15 8
      www.google.com www.armut.com
  www.bilkent.edu.tr 139.179.10.12 8 www.google.com 209.85.135.136 5
      www.xkcd.com www.milliyet.com.tr
  ```

```
www.fizy.com 85.17.82.33 8      www.youtube.com www.wikileaks.ch
www.radikal.com.tr www.google.com
```

- **Linear Hashing Structure Output File:** The output file where final linear hashing structure is printed. The file path is given as the forth parameter to your program.

- **Results of Queries File:** The file where the results of the coming queries are printed. The path is given as the fifth parameter to your program.

3. Output Specifications

- **Linear Hashing Structure Output:** Your program should print the final state of the linear hashing structure to the specified output file.

  Expected Linear Hashing Structure Output

```
www.bilkent.edu.tr    139.179.10.12    8    www.radikal.com.tr    213.243.16.14    5
www.wikileaks.ch 213.251.145.96 5     www.fizy.com          85.17.82.33          8
www.milliyet.com.tr 83.66.140.16 5
www.xkcd.com       72.26.203.99      8     www.google.com      209.85.135.136     5
www.yahoo.com 69.147.76.15 8
```

- **Results of Queries Output:** Your program should print the corresponding ip addresses of the queries. If there is no record for a query, just print -1.

  Expected Results of Queries Output File

```
208.65.153.238
69.63.184.142
216.239.51.99 -1
72.26.203.99 83.66.140.16
-1 213.251.145.96 213.243.16.14 209.85.135.136
```

4. Hash Function

```
public int getHashValue(String domainName)
  {
     int total=0;
     for(int i=0; i < domainName.length(); i++)
        if(Character.isLetter(domainName.charAt(i)))
           total+= i * (Character.getNumericValue(domainName.charAt(i)) -
Character.getNumericValue('a') + 1);
     return (total%10000);
  }
```

5. Sample Run Command for Your Program

java your-java-class 0.666 3 records-and-queries-file-path linear-hashing-file-structure-output-file-path results-of-queries-output-file-path

6. General Rules

- Your program must be named BasicDNSServer.java
- Your report must be named **StudentID.doc** (e.g. 20504152.doc)
- Note that any input file could be used for testing purposes. Also a supportive tool is used to detect plagiarism.

7. Tutorial: How to R/W File in JAVA

In JAVA, there are various ways to handle r/w a file; but you must use the following way. Other implementations will **not** be accepted.

You have to use BufferedReader and BufferedWriter classes to read and write a file respectively. Consider using the following methods of BufferedReader (You can also see JAVA API for all methods):

- *constructor*: **BufferedReader(Reader in):** Opens the file to read. The argument is an instance of Reader class. See JAVA API for more details.
- **readLine()**: Reads a line from the file.
- **close()**: Closes the file.

Also use the following methods of BufferedWriter:

- *constructor*: **BufferedWriter(Writer out):** Opens the file to write. The argument is an instance of Writer class. See JAVA API for more details.
- **write(String str):** Writes a string to the file.
- **close():** Closes the file.

8. Submission

I. Create a RAR archive file for your BasicDNSServer.java and 20504152.(doc | docx) files.

II. Name your RAR archive file as **StudentID_cs351p3.rar**(e.g. 20504152_cs351p3.rar). Files with other formats will be **ignored** automatically.

III. Use the submission page http://cs.bilkent.edu.tr/~ctoraman/cs351fall10/project3/submission/ to upload your RAR archive file. Read the upload rules in the submission page as well. Late submissions will **not** be accepted.

IV. You can resubmit your project until its due date.

V. For any question, contact with Emre VAROL(evarol@cs.bilkent.edu.tr).