COMPSCI 220 S1 C Assignment 3

Department of Computer Science

The University of Auckland

Due Monday, 13 May 2013, 9.00 *pm*

This assignment is worth 18 marks; it represents 6% of your total course grade.

Part 1: Distributed Hash Table Program [9 marks]

Write a program that models a distributed hash table (HT) as an array of 8 separate hash tables. Your DHT must implement the DHT's placement algorithm as a method:

```
Integer place(String key) { }
```

You should do this in Java using the SimpleHT class, which you can download from 220's Cecil Resources page.

SimpleHT is a simple-to-use class that implements a hash table using Java's Hashtable class, but that class has rather different methods to those discussed in lectures. SimpleHT implements a hash table, using a string as its key, and a single Integer as that key's value. [As you'll see from the source code, if the key isn't already in the table, we insert it with value 1; otherwise we increment the value by one, and put the new value back into the Hashtable.]

Your input data is a set of DNS records from the U Auckland network, showing requests from computers outside our network, and the (anonymised) Internet Address they came from. Use these two fields (i.e. the whole line) as the key for your DHT.

No value will be stored in each DHT record, however each record will contain an integer that is incremented by one each time its key appears in the file.

The input file has the following format:

```
Each record (line) has two blank-separated fields:
  domain_name source_address,  e.g.
student.auckland.ac.nz. 52.125.122.229
artsnet.auckland.ac.nz. 181.132.131.229
ec.auckland.ac.nz. 173.119.112.26
```

A small (100 records) sample file is available from 220's Cecil Resources page; download it and use it while developing your own hash table program.

N_ht.java, an example main() program, is provided (again, on the 220 Resources page). It demonstrates how you should read the data file, you may use this as a starting point for your own program.

CONTINUED

1. Write a program that reads the data file from standard input line by line, looks up the record for each line's key, and increment that record's count.
   At end-of-file, print out

   - the key that has the highest count
   - the number of times it appeared in the input data
   - the number of entries in each of the eight SimpleHTs (i.e. the SimpleHT sizes)
   - the maximum and minimum SimpleHT sizes
   - the difference between max and min SimpleHT sizes

   Test your program using the sample input data file;
   a sample output file for this is available on Cecil's 220 Resources page.
   Note: for marking, your program will be tested using a larger data file.

   Hint: you will need to implement a new "Distributed Hash Table" class, and may need to modify SimpleHT to provide the information in the list above.

   This is 'ModelDHT' for the automarker.

Part 2: Improving the placement algorithm [9 marks]

NOTE: this is a short-answer question!

1. For the simple placement function used in class DHT so far (hash = value of last byte of IPv4 address), hashing performance is poor – comment on what it is that suggests the performance is poor. [2 marks]
2. Change your code for DHT.place() to implement a better hash function, and run your modified program on the sample data file. Describe your improved algorithm. [4 marks]
3. Has hashing performance improved? By what percentage? Explain your answer. [3 marks]

   This is 'q2' for the automarker; automarker doesn't do anything with this file, it leaves that for the *human* markers) – it just shows a pale blue (other) box.

Questions involving programming

- You must use Java, and the SimpleHT class provided.
- Your answer to each question should be a single file (containing all nonstandard classes you use). You may assume that the markers have access to all standard libraries.
- A sample input file, and output files for each question will be available. The markers may check the output with a text comparison program, so it must be in EXACTLY the right format.
- The automated feedback and submission system ("automarker") is available. You must submit your answer via this system. You may take account of the feedback given by the automarker, and resubmit before the deadline without penalty. There is a limit of 15 submissions for each subproblem.
- Your program(s) may be tested on much larger input files. Marks will be allocated for correctness of the programs. Simply "passing" the automarker may not guarantee maximum marks, but it will guarantee full marks for correctness.

CONTINUED

Input and output format

For Q1, the following format will be used. Sample input and output is available on Cecil, the files are

- `dns-data-small.input`
- `dns-data-small.output`

Pay attention to line breaks and beware of nonstandard software. For best results, use a Linux environment such as login.cs (the automarker does).

The format of the input and output files is specified above, and is used for the sample input and output files. Remember, the automarker expects your output file to match the sample output *EXACTLY!*

Dates and Marks

This assignment is marked out of 27 and is worth 6% of your course grade. You should submit via the automarker:

- A source file for question 1. The name of this file must be 'ModelDHT.java' (i.e. the same as the automarker names of the subproblems).
- A plain text file called 'q2.txt' OR a PDF file called 'q2.pdf' containing the answers for question 2. No feedback will be given for these.

The due date is Monday, 13 May 2013, 9:00 pm.