**CSC2001F 2017 Assignment 1**

**Instructions**

Write an **Electronic Telephone Directory** application in Java, using a Binary Search Tree (BST) as an internal data structure.

Your BST implementation can be created from scratch or re-used from anywhere.  You may NOT replace the BST with a different data structure.

The data you are given (see attached file) has the following format:

51850 Kianna Squares, Terre Haute|552.531.3674|Gislason Kenna  
17386 Stephanie Parks, Palm Springs|018-594-2935 x716|Hickle Leone  
97354 Queen Squares, Birmingham|(332)985-4036|Moore Gilbert

The fields are: address, telephone number, name.  Every full name has a last name and a first name.  The fields are separated by "|".  The lines of the file are unsorted.

When loading this data, the key you must use is the full name.  You need to parse each line of text and extract the name.  Each record will therefore contain: {full\_name, full\_entry}.

Your tasks are as follows:

* Write an application **PrintIt** to load the data file into a BST and then traverse the BST and print out the telephone listing in order of name.  Where two people have the same names, the relative order of those entries does not matter.
* Write an application **SearchIt**to search for an entry based on a full name.  Your application must read in a list of queries from a query file, search for each one and output "Not found" or the full entry for each query.  The data file must be loaded only once.
* Write an application **SearchItLinear** with the same functionality as SearchIt, but replace your BST with an unsorted array.  Rewrite the necessary algorithms.
* Conduct an experiment with SearchIt and SearchItLinear to demonstrate the speed difference for searching between a BST and a linear array.  Extract names from the data file and use only these names as queries.  Use multiple subsets of the data from n=1 up to n=10000.  Draw a graph that illustrates the relative performance of your applications for different values of n.  Use the "time" Unix application to measure application execution time.  If your computer is too fast to take accurate readings of small amounts of time, run the experiment multiple times (possibly using a shell script).
* Write a report (of up to 10 pages) that includes the following:
  + What the problem is (at most half page).
  + The design of your application (at most 2 pages - only discuss PrintIt and SearchIt in detail).
  + The first 20 lines of the output from PrintIt.
  + A 20 entry query file used for SearchIt and SearchItLinear, followed by the output from each application when run with this query file.
  + The design of your comparison experiment, results, discussion and conclusions.
  + Summary statistics from your use of git and junit, to demonstrate usage and breadth of testing.

**Dev requirements**

As a software developer, you are required to make appropriate use of the following tools:

* git, for source code management
* junit, for unit testing
* javadoc, for documentation generation
* make, for automation of compilation, documentation generation, unit testing and cleaning of files

**Submission requirements**

Submit a .tar.gz compressed archive containing:

* Makefile
* src/
  + all source code
* bin/
  + all class files
* doc/
  + javadoc output
* test/
  + junit test classes
* report.pdf

Your report must be in PDF format.  Do not submit the junit jar files or the git repository.

**Marking Guidelines**

Your assignment will be marked by tutors, using the following marking guide.  You will be expected to demonstrate the application to your tutor to earn the "correctness" marks.  You can also explain to the tutor how you went beyond the stated requirements of the assignment to earn the "creativity" marks.

|  |  |  |
| --- | --- | --- |
| *Artefact* | *Aspect* | *Mark* |
| Report | Problem description | 5 |
|  | Application design | 10 |
|  | PrintIt output | 5 |
|  | SearchIt/SearchItLinear queries and output | 5 |
|  | Experimental design - complete and unambiguous | 5 |
|  | Results - tables, graphs | 5 |
|  | Discussion and conclusions | 5 |
|  | Statistics on tool usage - git - appropriate usage | 5 |
|  | Statistics on tool usage - junit - appropriate level of testing | 5 |
| Code | Correctness | 30 |
| Dev | Documentation - javadoc | 5 |
|  | Makefile - make, docs, junit and clean targets | 5 |
| Any | Creativity | 10 |