TREES

1. OBJECTIVES

(a) To review and strengthen the concept of trees.
(b) To practise the recursion techniques.
(c) To review different ways of traversing a binary tree.

2. LABORATORY

This lab will be conducted in the Computing Lab 1 (N4-B1-8) in SCE. This is an individual experiment. You cannot use the classes that implement the **Collection** interface (please refer to http://java.sun.com/docs/books/tutorial/collections/), except for Array class. We are going to implement these classes (like Vector, LinkedList, Stack, TreeSet, etc.) in CPE/CSC105, instead of using them directly from Java.

No make-up is allowed for students who have missed their stipulated lab classes without any acceptable excuse like having a valid leave of absence from the school or on medical leave. The students will be deemed to have failed the particular lab work. In case you have valid reasons to do makeup, you must inform the lecturer in charge. The makeup should be done within the same week, during the lab sessions attended by other groups.

3. EQUIPMENT

Hardware: The PCs running under the LINUX environment in Computing Lab 1.
Software: NetBeans, SUN JAVA compiler (**javac**) and interpreter (**java**). Your programming will only be tested by the lab markers using javac and java on the lab PCs.

4. EXPERIMENT

The *Bank of Datapore* has a set of customers who deposit or withdraw cash from their bank accounts regularly. Information about the customers is stored persistently as a text file on a hard disk in the bank. At the beginning of each month, all the customer records are read into the computer from the hard disk. Throughout the month, customers may deposit (credit) and/or withdraw (debit) from their accounts. This assignment handles:

- The input of customer records from a text file using Java.
- The storage of records into a BST.
- Searching for a record in a BST.
- Layer by layer tree traversal.

(a) A customer record contains the following attributes: a unique **accountid**, **name, address, date of birth, phone number**, and **account balance**. Declare a suitable class called **customer** to represent each customer record.

(b) Write a method **readFile** that reads customer records from a text file and stores them (according to the **accountid**) in a binary search tree (BST). No balancing is needed. The method reports as its return value and prints to standard output the number of customer records successfully read. In this context, you must declare two suitable classes to represent a BST node and a BST.

An example of a customer-record BST is shown in Figure 7.1. Its data in the text file format, **customers.txt**, can be found in Appendix A. In the tree, each node stores a customer record. The **key** for the BST is the **accountid** of each customer.

(c) Write a method **inOrder** that traverses the tree and prints the node values (print the customer ID only) in in-order manner.

(d) Write a method **deepestNodes** that finds and prints all the nodes at the deepest level. (**Hint**: employ layer by layer tree traversal)

(e) Write a method **descending** that traverses a BST tree and lists the node values in descending order manner. (**Hint**: change the conventional tree traversal to visit the right son first)

(f) Write a method **intersect** that takes the intersection of two trees. The returned value should be a reference to a newly created tree which holds the intersected result as shown in Figure 7.2, where the accountid of each intersected node is the sum of the corresponding nodes from tree1 and tree2, and the other values (i.e. names, address, etc.) are taken from the nodes in tree 1. (**Hint**: **intersect** should traverse tree1 and tree2 simultaneously)

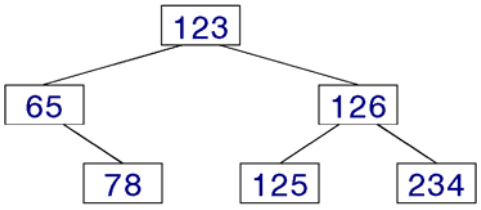

Figure 7.1: An example of a binary search tree.

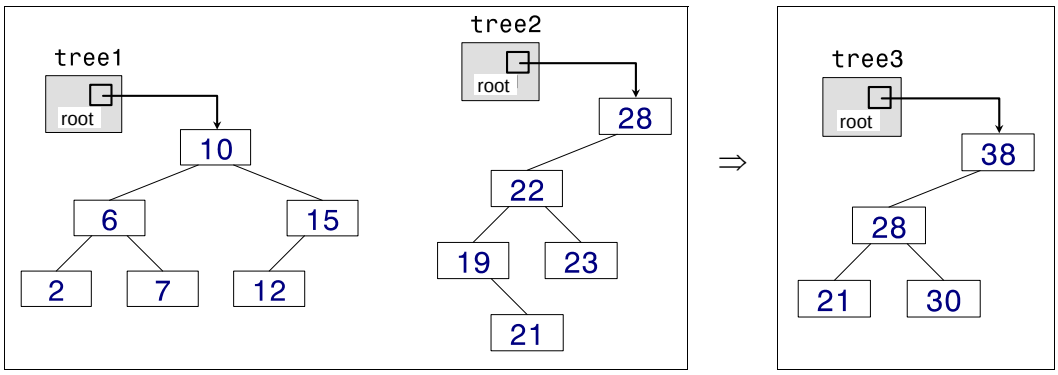

Figure 7.2: Taking the intersection of two trees.

(g) Write a **menu** that displays all the above methods (as shown below) for test run.

```
(1)   Read data to BST
(2)   In-order
(3)   Print deepest nodes
(4)   Print descending order
(5)   Intersect
(6)   Exit
```

Create 3 tree object, **t1**, **t2** and **t3**. When command 1 is chosen, the program should prompt for a file name to input the customer records into t1. When commands 2, 3 and 4 are chosen, the commands should be executed on t1. When command 5 is chosen, the program should prompt for 2 file names and input customer records into t1 and t2. The result should be stored in t3. Command 5 should be ended by printing the **post-order** listing of t3.

Although each of the above tasks is to be implemented as a single method, it may be necessary to create additional sub-method(s) to handle portions of the method. This is especially true if the original method is too long or contains more than one functionally related group of statements. When you write a method, remember that this method is to work for all possible inputs. Not on just your test inputs. You must test for all conditions that might possibly arise; print out error messages as needed.

A text file **customers.txt (Appendix A)** of customer records has been created in the directory **/home/staff1/csc105/lab7** to assist the development and testing of your program. Its content will be changed (while still adhering to the same data format) during grading in order to test the robustness of your program. The objective is to ensure that you do not hardcode your program to work exclusively for the given sample customer records. Check also the **readme** file for any last minute hints or changes.

Create a sub-directory called **cpe105/lab7** or **csc105/lab7** (depending on whether you are taking CPE105 or CSC105) in your home directory. Please note that the grader will only look into this sub-directory to find, compile and test run your program. You should provide a readme file that gives instructions to compile and run your program. You should not have irrelevant source files in the directory.

5. ERROR HANDLING

Your can assume that the input is always correct; thus no input validation is required.

6. REPORT

(a) Please check the posted final due date on Edventure for this lab.

(b) For the report, submit hard copies of the readme file, program listing and **script** file(s) recording results of the testing of your program.

(c) Not every lab report will be graded. You will be notified only after all the hard copies mentioned in part (b) have been received.

(d) The grading of your work will be based on the criteria listed in lab1 (c) and (d).

7. ACADEMIC HONESTY AND COLLABORATION

Cooperation is recommended in understanding various concepts and system features. But the actual solution of the assignments, the programming and debugging must be your individual work, except for what you specifically credit to other sources. (Your grade will be based on your own contribution.) For example, copying without attribution any part of someone else's program is plagiarism, even if you modify it and even if the source is a textbook. You can document the credit to other sources at the start of your program code listing. The University takes acts of cheating and plagiarism very seriously: first time violators may fail the coursework component of CPE105 / CSC105. Any wholly (or partly) copied (or being copied) programs will receive zero mark.

8. REFERENCES

[1] Text and reference books for CSC/CPE 105.

[2] http://java.sun.com/docs/index.html; http://java.sun.com/j2se/

Appendix A
customers.txt

Account Id = 123
Name = Matt Damon
Address = 465 Ripley Boulevard, Oscar Mansion, Singapore 7666322
DOB = 10-10-1970
Phone Number = 790-3233
Account Balance = 405600.00

Account Id = 126
Name = Ben Affleck
Address = 200 Hunting Street, Singapore 784563
DOB = 25-10-1968
Phone Number = 432-4579
Account Balance = 530045.00

Account Id = 65
Name = Salma Hayek
Address = 45 Mexican Boulevard, Hotel California, Singapore 467822
DOB = 06-04-73
Phone Number = 790-0000
Account Balance = 2345.00

Account Id = 78
Name = Phua Chu Kang
Address = 50 PCK Avenue, Singapore 639798
DOB = 11-08-64
Phone Number = 345-6780
Account Balance = 0.00

Account Id = 234
Name = Zoe Tay
Address = 100 Blue Eyed St, Singapore 456872
DOB = 15-02-68
Phone Number = 456-1234
Account Balance = 600.00

Account Id = 125
Name = Julia Roberts
Address = 100 Brokovich St, Singapore 456455
DOB = 15-08-65
Phone Number = 456-4321
Account Balance = 4567600.00