CS112 – Spring 2024

Lab22

Instructor: Paul Haskell

INTRODUCTION

In this lab, you will work with recursion and advanced data structures. This is another big lab…but the last big lab of the semester!

Euclid's Algorithm

Suppose you have two positive integers, A and B. The greatest common factor of A and B, denoted GCF(A, B), is the largest number that is a factor of both A and B. For example,

GCF(2, 4) = 2
GCF(4, 2) = 2
GCF(60, 45) = 15
GCF(50, 45) = 5

There is an algorithm credited to Euclid, the ancient Greek mathematician, for calculating the GCF of two numbers. Repeatedly invoke the following formula until it yields a final answer:

$$\text{GCF(A, B)} = \begin{cases} A, if\ B = 0 \\ GCF(B, A\ \%\ B), otherwise \end{cases}$$

Rather than trying to explain why this algorithm works, it will be more instructive for you to code it up and run it with several examples to see it in action. Please write a program called **Euclid.java** that implements the above recursive algorithm. The program shall input the numbers A and B as two separate command-line arguments, and it shall print out the GCF of A and B. Use our usual error handling rules: if there is a problem with the user input, print "ERROR" to `System.err`, along with an explanation of the error.

Binary Search

Suppose you are a software developer for the Chihuahua Action Network, a nonprofit that performs rescue work for those cute little dogs.

You are writing an information storage and retrieval system. Since so many people love Chihuahuas, the nonprofit has an enormous list of donors. Your retrieval software wants to search for particular donors quickly, not checking through every single entry in the donor list.

You will implement a technique called Binary Search. You will develop a program called **BinarySearch.java**. The donor list input to the program will be sent to `System.in` and will consist of multiple lines. Each line contains a donor's name. There may or not be a first name on every line, there may be one or more middle names, but there always will be a last name. The input will be sorted in increasing alphabetical order, for example:

Aaron Adams
Aaron Brown
Aaron Ziegler
Alicia Gutierrez
Brenda Fung
...

(You can check that the input is in alphabetical order, but this time the automatic grader won't care if you fail to detect it.)

BinarySearch.java will take a single command line argument, which will be a name. If the donor list input contains an exact match for the name, your program should print "MATCH". Otherwise it should print "NOT FOUND".

To search your file of names efficiently, you must use the "binary search" method, which works as follows:

- Check the middle entry in your name database. If the name you are searching for comes before the middle entry in alphabetical order, then continue searching only the first half of the database. If the name you are searching comes after the middle entry in alphabetical order, then continue searching only the second half of the database.
- Of course, if the middle entry matches the name you are searching for, you can return success
- If you search a subsection of the list with only 1 entry, and it does not match, then you can return failure

You can consider whether recursion would be useful for this problem. You are not required to use recursion.

Doubly Linked Ring

Now you will work on code for a doubly linked ring buffer. As we discussed in class, a doubly linked ring has links from each node to both previous and following nodes in the ring. **CourseInfo/Lab22** contains a program called **Ringbuf.java** with a partial implementation of a doubly linked ring. You will finish coding the missing methods. Don't forget to test!

Binary Tree

The **Lab22** directory in the CourseInfo repository contains a Java file called **TestTree.java** that builds a binary tree and fills it with values read from a file (which contains integers separated by spaces). The filename is given in `args[0]`. Your job is to finish the code for several methods, some of which should be recursive.

- One method adds an integer value to the tree.

- One method adds up all of the integer values stored in the tree and returns the result
- One method returns the minimum (most negative) value in the tree
- One method returns the size (number of entries) in the tree
- One method returns the "maximum depth" of the tree

Reminder

Put all your files in your **Lab22** directory and push to GitHub before the deadlines.

- **Euclid.java** and **BinarySearch.java** are due at the end of class
- **Ringbuf.java** and **TestTree.java** must be turned in before 11:59pm Monday April 22.

Conclusion

The experience from this week's lab with recursion will help with the course's second large project.

Grading Rubric

Euclid.java is worth 10 points: 2 points for each of 5 test cases

BinarySearch.java is worth 20 points: 3 points for each of 5 test cases, plus 0-5 points for software quality. Zero points overall if you don't use a binary search!

Ringbuf.java is worth 15 points: 10 points for correct output, 5 points for code quality, including software design quality, code clarity, comments, and correctness of code.

TestTree.java is worth 20 points: 16 points if it compiles, runs, and gives proper output with four test cases. 0-4 points for code quality, including software design quality, code clarity, comments, and correctness of code.