Data Structures and Algorithms in Java™

Sixth Edition

Michael T. Goodrich
Department of Computer Science
University of California, Irvine

Roberto Tamassia
Department of Computer Science
Brown University

Michael H. Goldwasser
Department of Mathematics and Computer Science
Saint Louis University

Study Guide: Hints to Exercises

WILEY

Chapter 1 Java Primer

Hints

Reinforcement

R-1.1) Use the code templates provided in the Simple Input and Output section.

R-1.2) You may read about cloning in Section 3.6.

R-1.3) The modulus operator could be useful here.

R-1.4) Use bit operations.

R-1.5) The easy solution uses a loop, but there is also a formula for this, which is discussed in Chapter 4.

R-1.6) The easy thing to do is to write a loop.

R-1.7) The easy thing to do is to write a loop.

R-1.8) You might use a switch statement.

R-1.9) Consider each character one at a time.

R-1.10) Consider using get and set methods for accessing and modifying the values.

R-1.11) The traditional way to do this is to use setFoo methods, where Foo is the value to be modified.

R-1.12) Use a conditional statement.

R-1.13) Try to make wallet[1] go over its limit.

Creativity

C-1.14) The Java method does not need to be passed the value of n as an argument.

C-1.15) Note that the Java program has a lot more syntax requirements.

C-1.16) Create an enum type of all operators, including $=$, and use an array of these types in a switch statement nested inside for-loops to try all possibilities.

C-1.17) Note that at least one of the numbers in the pair must be even.

C-1.18) Use the Math.pow function for calculations. Use your solution for norm(v,p) to implement norm(v).

C-1.19) This is the same as the logarithm, but you can use recursion here rather than calling the log function.

C-1.20) The simple solution just checks each number against every other one, but we will discuss better solutions later in the book. Make sure you don't compare a number to itself.

C-1.21) Consider using swaps to reshuffle the array one entry at a time, starting from the beginning and moving to the end.

C-1.22) There are many solutions. If you know about recursion, the easiest solution uses this technique. Otherwise, consider using an array to hold solutions. If this still seems to hard, then consider using six nested loops (but avoid repeating characters and make sure you allow all string lengths).

C-1.23) Go back to the definition of dot product and write a for loop that matches it.

C-1.24) The card is no longer needed as an explicit parameter.

C-1.25) You might use a StringBuilder to compose the pieces of the string into one large string (including newlines).

Projects

P-1.26) Use an array to buffer all the original lines.

P-1.27) You do not need to use a graphical user interface, but you may want to use the System.console() method.

P-1.28) Define a way of indexing all the sentences and the location in each one and then work out a way of picking eight of these locations for a typo.

P-1.29) Use a two-dimensional array to keep track of the statistics and a one-dimensional array for each experiment.

P-1.30) We recommend using the Java Swing package.