Exercises and Homework

java.util Methods for Arrays

fill(A, x)

copyOf(A, n)

copyOfRange(A, s, t):

toString(A)

sort(A):

binarySearch(A, x)

|  |  |  |
| --- | --- | --- |
| 1 | R-3.1 | Give the next five pseudorandom numbers generated by the process described on page 113, with a = 12, b = 5, and n = 100, and 92 as the seed for cur.  See page 113  next = (a \* cur + b) % n;  = |
| 2 | R-3.2 | Write a Java method that repeatedly selects and removes a random entry from an array until the array holds no more entries. |
| 3 | R-3.3 | Explain the changes that would have to be made to the program of Code Fragment 3.8 so that it could perform the Caesar cipher for messages that are written in an alphabet-based language other than English, such as Greek, Russian, or Hebrew.  Use a different character encoding system that supports the desired language, such as UTF-8 or Unicode. This will allow us to represent letters in the language as numbers that can be manipulated by a cryptographic algorithm |
| 4 | R-3.4 | The TicTacToe class of Code Fragments 3.9 and 3.10 has a flaw, in that it allows a player to place a mark even after the game has already been won by someone. Modify the class so that the putMark method throws an IllegalStateException in that case  It is to add a condition to check if the game has already been won or not |
| 5 | R-3.13 | What is the difference between a shallow equality test and a deep equality test between two Java arrays, A and B, if they are one-dimensional arrays of type int? What if the arrays are two-dimensional arrays of type int?    The difference is How to compare elements within arrays.. Deep equality test examines the values of elements within two arrays This means that it checks whether every element at the same index in both arrays is equal  For matrices with multiple dimensions, testing becomes more complex and requires comparing element values within each subarray. |
| 6 | R-3.14 | Give three different examples of a single Java statement that assigns variable, backup, to a new array with copies of all int entries of an existing array, original.   1. Use the clone method to clone an array. 2. Use arraycopy() method of System class. 3. Use copyOf() or copyOfRange() methods of Arrays class.   int[] original = {1, 2, 3, 4, 5};  int[] backup = new int[original.length]; System.arraycopy(original, 0, backup, 0, original.length);  int[] original = {1, 2, 3, 4, 5};  int[] backup = Arrays.copyOf(original, original.length);  int[] original = {1, 2, 3, 4, 5};  int[] backup = new int[original.length]; for (int i = 0; i < original.length; i++) {  backup[i] = original[i];  } |
| 7 | C-3.17 | Let A be an array of size n ≥ 2 containing integers from 1 to n−1 inclusive, one of which is repeated. Describe an algorithm for finding the integer in A that is repeated. |
| 8 | C-3.18 | Let B be an array of size n ≥ 6 containing integers from 1 to n−5 inclusive, five of which are repeated. Describe an algorithm for finding the five integers in B that are repeated.  **Algorithm:**   1. Create a set S to store the distinct elements encountered so far. Initialize S to an empty set. 2. Iterate through the array B: a. For each element b in B: i. If b is not in S, add b to S. This indicates that the element b has been seen once. ii. If b is already in S, then b is a repeated element. Add b to a list of repeated elements. 3. Since there are five repeated elements, continue iterating through B until you find five distinct elements that are repeated. 4. The list of repeated elements contains the five repeated integers in B.   **Analysis:**  Time Complexity: O(n), where n is the size of the array B. This is because the algorithm iterates through the array B only once, and each operation takes constant time.  Space Complexity: O(n-5), where n-5 is the size of the set S. This is |
| 9 | C-3.19 | Give Java code for performing add(e) and remove(i) methods for the Scoreboard class, as in Code Fragments 3.3 and 3.4, except this time, don’t maintain the game entries in order. Assume that we still need to keep n entries stored in indices 0 to n−1. You should be able to implement the methods without using any loops, so that the number of steps they perform does not depend on n. |
| 10 | C-3.20 | Give examples of values for a and b in the pseudorandom generator given on page 113 of this chapter such that the result is not very random looking, for n = 1000.  A=0  B=0 |
| 11 | C-3.21 | Suppose you are given an array, A, containing 100 integers that were generated using the method r.nextInt(10), where r is an object of type java.util.Random. Let x denote the product of the integers in A. There is a single number that x will equal with probability at least 0.99. What is that number and what is a formula describing the probability that x is equal to that number?  we need to analyze the possible values of x, the product of the integers in array A. Since each integer in A is generated using the method r.nextInt(10), it can take values from 0 to 9 inclusive.  The probability that x equals a specific number n can be calculated by considering the number of ways we can obtain n as the product of the integers in A, divided by the total number of possible outcomes. |
| 12 | C-3.22 | Write a method, shuffle(A), that rearranges the elements of array A so that every possible ordering is equally likely. You may rely on the nextInt(n) method of the java.util.Random class, which returns a random number between 0 and n−1 inclusive. |
| 13 | C-3.23 | Suppose you are designing a multiplayer game that has n ≥ 1000 players, numbered 1 to n, interacting in an enchanted forest. The winner of this game is the first player who can meet all the other players at least once (ties are allowed). Assuming that there is a method meet(i, j), which is called each time a player i meets a player j (with i 6= j), describe a way to keep track of the pairs of meeting players and who is the winner.  each player is represented by a Player object, which keeps track of the other players they have met using a Set<Integer> called metPlayers. The meet method is responsible for updating the meeting records for both players involved and checking if the current player is the winner. The Game class manages the game logic and maintains a map of player IDs to Player objects.  The game loop simulates all possible meetings between players. When a player meets another player, the meet method is called, updating the meeting records for both players and checking if the current player is the winner by calling hasMetAllPlayers method |
| 14 | C-3.24 | Write a Java method that takes two three-dimensional integer arrays and adds them componentwise.  }  return result;  } |