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  الأعداد الخمسة الشبه عشوائية التالية هي: 99، 93، 21، 57، و 89 |
| 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.  import java.util.Arrays;  import java.util.Random;  public class RandomEntryRemoval {  public static void main(String[] args) {  String[] array = {"العنصر 1", "العنصر 2", "العنصر 3", "العنصر 4", "العنصر 5"};  System.out.println("المصفوفة الأصلية: " + Arrays.toString(array));  removeRandomEntries(array);  System.out.println("المصفوفة بعد الإزالة: " + Arrays.toString(array));  }  public static void removeRandomEntries(Object[] array) {  Random random = new Random();  for (int i = array.length - 1; i > 0; i--) {  int randomIndex = random.nextInt(i + 1);    Object temp = array[i];  array[i] = array[randomIndex];  array[randomIndex] = temp;    array[i] = null;  }  }  } |
| 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.  1. مجموعة الأحرف  2. تعيين الأحرف  3. التعامل مع حساسية الحالة |
| 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  public class TicTacToe {    public void putMark(int row, int col) {  if (isGameOver()) {  throw new IllegalStateException("تم الفوز بالفعل في اللعبة.");  }  if (isWinningMove(row, col)) {  gameOver = true;  winner = currentPlayer;  } else {  currentPlayer = (currentPlayer == Mark.X) ? Mark.O : Mark.X;  }  }  private boolean isGameOver() {  return gameOver;  }  private boolean isWinningMove(int row, int col) {    }  } |
| 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?  الاختبار السطحي للمساواة (shallow equality test) يقارن المراجع (references) للمصفوفتين، بينما الاختبار العميق للمساواة (deep equality test) يقارن المحتوى الفعلي لكل خلية في المصفوفتين. |
| 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. استخدام حلقة for:  java  int[] backup = new int[original.length];  for (int i = 0; i < original.length; i++) {  if (original[i] instanceof Integer) {  backup[i] = original[i];  }  }  2. استخدام تعبير Lambda وتحويل المصفوفة:  java  int[] backup = Arrays.stream(original)  .filter(element -> element instanceof Integer)  .mapToInt(element -> (int) element)  .toArray();  3. استخدام حلقة for-each:  ```java  int[] backup = new int[original.length];  int index = 0;  for (Object element : original) {  if (element instanceof Integer) {  backup[index++] = (int) element;  }  } |
| 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.  إنشاء مجموعة Set فارغة.  قم بتكرار العناصر في المصفوفة A..  إذا تم انتهاء الحلقة دون العثور على العدد المكرر، فهذا يشير إلى وجود خطأ في المدخلات. |
| 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.  RepeatedIntegers(B):  n = طول المصفوفة B  hashTable = جدول تجزئة فارغ أو قاموس فارغ  repeatedIntegers = قائمة فارغة    لكل i من 0 إلى n-1:  إذا لم يكن B[i] موجودًا في hashTable:  قم بإضافة B[i] إلى hashTable بتكرار واحد  آخرى:  قم بزيادة قيمة B[i] في hashTable بمقدار واحد    لكل مفتاح في hashTable:  إذا كانت قيمة المفتاح في hashTable أكبر من واحد:  قم بإضافة المفتاح إلى قائمة الأعداد المتكررة    أعد repeatedIntegerz |
| 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.  public class Scoreboard {  private int numEntries = 0;  private GameEntry[] board;  public Scoreboard(int capacity) {  board = new GameEntry[capacity];  }  public void add(GameEntry entry) {  if (numEntries < board.length || entry.getScore() > board[numEntries - 1].getScore()) {  if (numEntries < board.length) {  numEntries++;  }  int i = numEntries - 1;  while (i > 0 && entry.getScore() > board[i - 1].getScore()) {  board[i] = board[i - 1];  i--;  }  board[i] = entry;  }  }  public GameEntry remove(int index) {  if (index < 0 || index >= numEntries) {  throw new IndexOutOfBoundsException("Invalid index: " + index);  }  GameEntry removedEntry = board[index];  for (int i = index; i < numEntries - 1; i++) {  board[i] = board[i + 1];  }  board[numEntries - 1] = null;  numEntries--;  return removedEntry;  }  } |
| 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.  1. قيمة a = 1 وقيمة b = 0:  في هذه الحالة، ستتكرر النتائج بشكل مستمر وسيكون لدينا تسلسل ثابت من الأعداد.  2. قيمة a = 2 وقيمة b = 3:  في هذه الحالة، سيتم إنتاج تسلسل من الأعداد الذي يتكرر بنمط ثابت. على سبيل المثال، النتائج قد تكون 3، 5، 7، 9، 11، 13، وهكذا.  3. قيمة a = 0 وقيمة b = 1:  في هذه الحالة، ستكون جميع النتائج متسلسلة بشكل متتالي، مثل 1، 2، 3، 4، 5، وهكذا. |
| 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?  P(x = 0) = 0.99  P(x ≠ 0) = 0.1 |
| 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.  import java.util.Random;  public class ShuffleArray {  private static Random rand = new Random();  public static void shuffle(int[] A) {  int n = A.length;  for (int i = n - 1; i > 0; i--) {  int j = rand.nextInt(i + 1);  swap(A, i, j);  }  }  private static void swap(int[] A, int i, int j) {  int temp = A[i];  A[i] = A[j];  A[j] = temp;  }  public static void main(String[] args) {  int[] A = {1, 2, 3, 4, 5};  shuffle(A);  for (int num : A) {  System.out.print(num + " ");  }  }  } |
| 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.  public class MultiplayerGame {  private String[][] meetings;  private int winner;  public MultiplayerGame(int numPlayers) {  meetings = new String[numPlayers][numPlayers];  winner = -1;  for (int i = 0; i < numPlayers; i++) {  for (int j = 0; j < numPlayers; j++) {  if (i != j) {  meetings[i][j] = "No";  }  }  }  }  public void meet(int player1, int player2) {  meetings[player1 - 1][player2 - 1] = "Yes";  meetings[player2 - 1][player1 - 1] = "Yes";  checkWinner(player1);  checkWinner(player2);  }  private void checkWinner(int player) {  for (int i = 0; i < meetings.length; i++) {  if (i != (player - 1) && meetings[player - 1][i].equals("No")) {  return;  }  }  winner = player;  }  public int getWinner() {  return winner;  }  public static void main(String[] args) {  int numPlayers = 1000;  MultiplayerGame game = new MultiplayerGame(numPlayers);  for (int i = 1; i <= numPlayers; i++) {  for (int j = i + 1; j <= numPlayers; j++) {  game.meet(i, j);  }  }  System.out.println("The winner is player: " + game.getWinner());  }  } |
| 14 | C-3.24 | Write a Java method that takes two three-dimensional integer arrays and adds them componentwise.  }  return result;  } |