## 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.<br>See page 113 |
| 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.ArrayList;
import java.util.Random;

public class RandomEntryRemoval {
    public static void removeRandomEntries(Object[] array) {
        ArrayList<Object> list = new ArrayList<>(array.length);

        for (Object entry : array) {
            list.add(entry);
        }

        Random random = new Random();

        while (!list.isEmpty()) {
            int randomIndex = random.nextInt(list.size());
            Object removedEntry = list.remove(randomIndex);
 System.out.println("الإدخال إزالة تمت: " + removedEntry);
        }
    }
    public static void main(String[] args) {
        String[] fruits = {"تفاح", "موز", "برتقال", "مانجا",
"أناناس"};
        RandomEntryRemoval.removeRandomEntries(fruits);
    }
}
```

| | | |
|---|---|---|
| | | |
| 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.<br><br>1. تعديل الحجم الثابت لمصفوفة الأبجدية الإنجليزية المستخدمة في البرنامج لتشمل الأبجدية المناسبة للغة المستهدفة. مثلاً، للتشفير بلغة يونانية، يجب تضمين الأحرف اليونانية في المصفوفة.<br>2. تعديل الإشارات المرجعية لكل حرف في المصفوفة لتتطابق مع الأبجدية المستهدفة. يجب أن يكون لكل حرف قيمة رقمية محددة تمثل تأثير التشفير عليه.<br>3. ضمان إمكانية التعامل مع حالات الأحرف الكبيرة والصغيرة بشكل صحيح. في البرنامج الأصلي، يتم استخدام الدوال `Character.isUpperCase()` و `Character.isLowerCase()` للتحقق من حالة الحرف. يجب التأكد من أن هذه الدوال متوافقة مع الأبجدية المستهدفة.<br>4. التأكد من إجراء التشفير بشكل صحيح. في البرنامج الأصلي، يتم تحويل رقم الحرف إلى الموقع النسبي للحرف في المصفوفة ، ومن ثم يتم تطبيق الإزاحة `charValue - 'A'` باستخدام العملية عليه. يجب التأكد من أن هذه العملية تعمل بشكل صحيح مع الأبجدية المستهدفة.<br>5. التأكد من التعامل السليم مع الحروف غير المشفّرة، بحيث لا يتم تطبيق التشفير عليها. في البرنامج الأصلي، يتم تجاهل الحروف غير الأبجدية الإنجليزية. يجب التأكد من أن البرنامج يتعامل بنفس الطريقة مع الحروف غير المشفّرة في الأبجدية المستهدفة. |
| 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<br>[redacted] |

| | | |
|---|---|---|
| | | <table><tr><td><pre>public void putMark(int row, int col) {<br>    if (board[row][col] != EMPTY) {<br>        throw new IllegalArgumentException("Cell already occupied!");<br>    }<br><br>    if (isGameOver()) {<br>        throw new IllegalStateException("Game is already over!");<br>    }<br><br>    board[row][col] = currentPlayer;<br>    currentPlayer = (currentPlayer == X_MARK) ? O_MARK : X_MARK;<br>}</pre></td></tr></table> |
| 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?<br><br>هناك فرق بين اختبار المساواة الضحلة واختبار المساواة في Java ، العميقة بين مصفوفتين.<br>وأحادية البعد، فاختبار المساواة int من نوع B و A إذا كانت المصفوفتين سيقوم بمقارنة عناوين ذاكرة المصفوفتين. يعني ذلك أنه (==) الضحلة تشيران إلى نفس المصفوفة، فسيعتبر الاختبار متساويًا. B و A إذا كانت تشيران إلى مصفوفتين مختلفتين حتى وإن كانتا B و A ولكن إذا كانت تحتويان على نفس القيم، فسيعتبر الاختبار غير متساوي.<br>، فإن اختبار المساواة int وبالنسبة للمصفوفات ثنائية الأبعاد من نوع سيقوم بمقارنة عنوان الذاكرة الأولي لكل مصفوفة. ولكنه (==) الضحلة B و A لن يقارن قيمة كل خلية في المصفوفة. بمعنى آخر، إذا كانت تشيران إلى نفس العنوان الأولي للمصفوفة، فسيعتبر الاختبار متساويًا. تشيران إلى عناوين أولية مختلفة حتى وإن كانتا B و A ولكن إذا كانت تحتويان على نفس القيم في كل خلية، فسيعتبر الاختبار غير متساوي.<br>إذا كنت بحاجة إلى مقارنة قيم كل خلية في المصفوفة، فيجب عليك الذي يقوم (`Arrays.deepEquals()`) استخدام اختبار المساواة العميقة بمقارنة القيم داخل المصفوفة. سواءً كانت المصفوفة أحادية البعد أو ثنائية الأبعاد، فإن اختبار المساواة العميقة سيقوم بمقارنة قيم كل خلية |

| | | |
|---|---|---|
| | | في المصفوفة وسيُعتبر المصفوفتين متساويتين إذا كانت قيم كافة الخلايا متطابقة. |
| 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.<br>1. استخدام حلقة for للنسخ:<br>`int[] originalArray = {1, 2, 3, 4, 5};`<br>`int[] backupArray = new int[originalArray.length];`<br>`for (int i = 0; i < originalArray.length; i++) {`<br>`backupArray[i] = originalArray[i];`<br>`}`<br>2. استخدام الدالة `System.arraycopy()`:<br>`int[] originalArray = {1, 2, 3, 4, 5};`<br>`int[] backupArray = new int[originalArray.length];`<br>`System.arraycopy(originalArray, 0, backupArray, 0, originalArray.length);`<br>3. استخدام الدالة `Arrays.copyOf()`:<br>`import java.util.Arrays;`<br>`int[] originalArray = {1, 2, 3, 4, 5};`<br>`int[] backupArray = Arrays.copyOf(originalArray, originalArray.length);` |
| 7 | C-3.17 | Let A be an array of size $n \geq 2$ 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.<br>`import java.util.HashSet;`<br>`import java.util.Set;`<br>`public class FindDuplicateNumber {`<br>`public static int findDuplicate(int[] nums) {`<br>`Set<Integer> set = new HashSet<>();`<br>`for (int num : nums) {`<br>`if (set.contains(num)) {` |

<table>
<tr><td></td><td></td><td>

```
                return num;
            }
            set.add(num);
        }

        throw new IllegalArgumentException("No duplicate number found!
");
    }
}
```

</td></tr>
<tr><td>8</td><td>C-3.18</td><td>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.

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class FindDuplicateNumbers {
    public static List<Integer> findDuplicates(int[] nums) {
        Set<Integer> set = new HashSet<>();
        List<Integer> duplicates = new ArrayList<>();

        for (int num : nums) {
            if (set.contains(num)) {
                duplicates.add(num);
            }
            set.add(num);
        }

        return duplicates;
    }
}
```

</td></tr>
</table>

| | | |
|---|---|---|
| | | |
| 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. |

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Scoreboard {
    private Map<String, Integer> scores;
    private List<String> players;

    public Scoreboard() {
        scores = new HashMap<>();
        players = new ArrayList<>();
    }

    public void addEntry(String player, int score) {
        scores.put(player, score);
        players.add(player);
    }

    public void removeEntry(String player) {
```

| | | |
|---|---|---|
| | | <br>`scores.remove(player);`<br>`players.remove(player);`<br>`}`<br>`}` |
| 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. |
| 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? |
| 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. |

```
public class MultiplyProbability {
    public static void main(String[] args) {
        Random r = new Random();
        int[] A = new int[100];
        int product = 1;

        for (int i = 0; i < A.length; i++) {
            A[i] = r.nextInt(10);
            product *= A[i];
        }

        System.out.println("Product: " + product);
        double probability = 1 - Math.pow((1 - (1.0 / 10)), A.length);
        System.out.println("Probability: " + probability);
    }
}
```

```
import java.util.Random;

public class ShuffleArray {
    public static void shuffle(int[] A) {
```

<table>
<tr><td></td><td></td><td>

```
        Random random = new Random();

        for (int i = A.length - 1; i > 0; i--) {
            int j = random.nextInt(i + 1);
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;
        }
    }
}
```

</td></tr>
<tr><td>13</td><td>C-3.23</td><td>

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.

للعثور على الفائز في اللعبة، يمكننا استخدام الخوارزمية التالية:

1. قم بتهيئة المصفوفة encounters بالقيمة 0.
2. قم بتكرار الخطوات التالية حتى يتم العثور على الفائز:
   - اختر عدد عشوائي بين 1 و n وسمه currentPlayer.
   - قم بإعلان winner واجعله يساوي currentPlayer.
   - قم بتكرار من 1 إلى n:
     - اختر عدد عشوائي بين 1 و n وسمه opponent.
     - إذا كان opponent لا يساوي currentPlayer وقيمة العنصر (currentPlayer, opponent) في encounters هي 0، فقم بتحديث encounters المصفوفة واجعل العنصر (currentPlayer, opponent) والعنصر (opponent, currentPlayer) يساوي 1 يساوي -1.
   - قم بفحص المصفوفة encounters للتحقق مما إذا كان currentPlayer قد التقى بجميع اللاعبين الآخرين. إذا كان اللاعب currentPlayer قد التقى

</td></tr>
</table>

| | | |
|---|---|---|
| | | واخرج currentPlayer إلى winner بالجميع، قم بتعيين<br>من الحلقة.<br>على السجلات encounters بعد انتهاء الحلقة السابقة، ستحتوي المصفوفة التي توضح اللاعبين الذين التقوا مع بعضهم البعض. وسيتم تحديد الفائز كلاعب يكون قد التقى بجميع اللاعبين الآخرين ولم يكن هناك لاعب يكون قد التقى به مرة أخرى. |
| 14 | C-3.24 | Write a Java method that takes two three-dimensional integer arrays and adds them componentwise.<br>`public class MatrixAddition {`<br>`  public static int[][][] addMatrices(int[][][] matrix1, int[][][] matrix2) {`<br>`  int rows = matrix1.length;`<br>`  int columns = matrix1[0].length;`<br>`  int depth = matrix1[0][0].length;`<br>`  int[][][] result = new int[rows][columns][depth];`<br>`  for (int i = 0; i < rows; i++) {`<br>`  for (int j = 0; j < columns; j++) {`<br>`  for (int k = 0; k < depth; k++) {`<br>`  result[i][j][k] = matrix1[i][j][k] + matrix2[i][j][k];`<br>`  }`<br>`  }`<br>`  }`<br>`  return result;`<br>`  }`<br>`}` |