**Data Structure Lab3 -Arrays**

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 113x

9

13

61

37

49

لحساب الأرقام الخمسة التالية باستخدام المولد الخطي التوافقي، نبدأ من الرقم 49 (آخر رقم في السلسلة المعطاة) ونطبق الصيغة:

cur\_next= (a .cur + b) mod n

1. الرقم 49:

cur\_1 = (12 .49 + 5) mod 100 = (588 + 5) mod 100 = 593 mod 100 = 93

2. الرقم 93:

cur\_2 = (12 .93 + 5) mod 100 = (1116 + 5) mod 100 = 1121 mod 100 = 21

3. الرقم 21:

cur\_3 = (12 .21 + 5) mod 100 = (252 + 5) mod 100 = 257mod 100 = 57

4. الرقم 57:

cur\_4 = (12 . 57 + 5) mod 100 = (684 + 5) mod 100 = 689 mod 100 = 89

5. الرقم 89:

cur\_5 = (12 . 89 + 5) mod 100 = (1068 + 5) mod 100 = 1073 mod 100 = 73

إذن، سلسلة الأرقام بالكامل تكون: 9، 13، 61، 37، 49، 93، 21، 57، 89، 73

2-Write a Java method that repeatedly selects and removes a random entry from an array until the array holds no more entries.

public class RandomRemover {

public static void removeRandomEntries(int[] array) {

Random random = new Random();

while (array.length > 0) {

int index = random.nextInt(array.length);

System.out.println("تم حذف العنصر: " + array[index]);

for (int i = index; i < array.length - 1; i++) {

array[i] = array[i + 1];

}

array = Arrays.copyOf(array, array.length - 1);

}

System.out.println("المصفوفة الآن فارغة");

}

public static void main(String[] args) {

int[] array = {1, 2, 3, 4, 5};

removeRandomEntries(array);

}

}

3-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 void putMark(int i, int j) throws IllegalArgumentException, IllegalStateException {

// التحقق إذا كانت اللعبة قد انتهت

if (winner() != 0) {

throw new IllegalStateException("لا يمكنك وضع علامة لأن اللعبة انتهت بالفعل");

}

// التحقق من صلاحية الموقع على اللوحة

if ((i < 0) || (i > 2) || (j < 0) || (j > 2)) {

throw new IllegalArgumentException("موقع غير صالح على اللوحة");

}

// التحقق إذا كانت الخلية مشغولة

if (board[i][j] != EMPTY) {

throw new IllegalArgumentException("الخلية مشغولة بالفعل");

}

// وضع العلامة الحالية وتبديل اللاعب

board[i][j] = player;

player = -player;

}

# 4-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. استبدال الأبجدية الإنجليزية بأبجدية ديناميكية:

المشكلة الأصلية: البرنامج يفترض وجود الأبجدية الإنجليزية (26 حرفًا من 'A' إلى 'Z') بشكل ثابت.

التعديل: تعريف الأبجدية كمدخل من المستخدم، يتم تمريرها كسلسلة نصية (String) تحتوي على جميع الأحرف الخاصة باللغة الجديدة.

2. التعامل مع عدد مختلف من الأحرف:

المشكلة الأصلية: البرنامج يعتمد على عدد ثابت من الأحرف (26 حرفًا).

التعديل: يتم حساب عدد الأحرف ديناميكيًا باستخدام alphabet.length()، ما يسمح بدعم أبجديات ذات أحجام مختلفة (مثل 24 حرفًا في اليونانية أو 33 في الروسية).

3. تحديد موقع الأحرف داخل الأبجدية:

المشكلة الأصلية: البرنامج يستخدم القيم الرقمية للحروف (ASCII) لتحديد مواقعها.

التعديل: استخدمنا دالة( alphabet.indexOf(char لتحديد موضع الحرف داخل الأبجدية الجديدة، مما يجعل البرنامج مناسبًا لأي أبجدية.

package prog;

class CaesarCipher {

protected char[] encoder;

protected char[] decoder;

protected String alphabet;

public CaesarCipher(String alphabet, int rotation) {

this.alphabet = alphabet;

int n = alphabet.length();

encoder = new char[n];

decoder = new char[n];

for (int k = 0; k < n; k++) {

encoder[k] = alphabet.charAt((k + rotation) % n);

decoder[k] = alphabet.charAt((k - rotation + n) % n);

}

}

public String encrypt(String message) {

return transform(message, encoder);

}

public String decrypt(String secret) {

return transform(secret, decoder);

}

private String transform(String original, char[] code) {

char[] msg = original.toCharArray();

for (int k = 0; k < msg.length; k++) {

int index = alphabet.indexOf(msg[k]);

if (index != -1) {

msg[k] = code[index];

}

}

return new String(msg);

}

}

public class NewClass {

public static void main(String[] args) {

// مثال باستخدام الأبجدية اليونانية

String greekAlphabet = "ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ";

CaesarCipher cipher = new CaesarCipher(greekAlphabet, 3);

System.out.println("شيفرة التشفير = " + new String(cipher.encoder));

System.out.println("شيفرة فك التشفير = " + new String(cipher.decoder));

String message = "ΗΕΛΛΟ";

String coded = cipher.encrypt(message);

System.out.println("النص المشفر: " + coded);

String answer = cipher.decrypt(coded);

System.out.println("النص الأصلي: " + answer);

}

}

5-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?

1-المصفوفات احادية البعد:

-الاختمار السطحي :يتم مقارنة المراجع الخاصة بالمصفوفتين باستخدام معامل ==

-الاختبار العميق: يتم مقارنة محتويات المصفوفتين عنصراً عنصر يتم استخدام ()Arrays.equals

2-المصفوفات ثنائية البعد:

-الاختبار السطحي:نفس المصفوفة احادية البعد يتم مقارنة المراجع

-الاختبار العميق:يتم مقارنة المحتويات الداخلية للمصفوفات بما في ذلك العناصر الفرعية

يتم باستخدام ()Arrays.deeoEquals

6-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.

backup = original.clone( );

int[ ] temp = Arrays.copyOf(original, n);

**public** **static** **void** arraycopy(

    Object src\_array, **int** src\_Pos,Object dest\_array, **int** dest\_Pos, **int** length )

System.arraycopy(src\_array, 0, dest\_array, 0,19);

الاجوبة

الطريقة الاولى

int[] backup=original.clone():

الطريقة الثانية

int[] backup=Arrays.copyOf(original ,original.length);

الطريقة الثالثة

int[] backup=new int[original.length];

; (System.copyOf(original, 0,backup,0,original.length

7-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.

def find\_repeated\_element(B):

distinct\_elements = set();

for b in B:

if b in distinct\_elements:

return b;

else:

distinct\_elements.add(b);

return None;

public static int findRepeatedElement(int[] A){

HashSet<Integer> distincetElement = new HashSet<>();

for (int b : A){

if (distincetElement.contains(b)){

return b;}

else {

distincetElement.add(b);

}}

return -1;

}

public static void main(String[] args){

int[] A = {5, 3, 2, 1};

System.out.println(findRepeatedElement(A));

}

8-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 the set S stores a maximum of n-5 distinct elements.

public class FindRepeatedElements {

public static List<Integer> findRepeated(int[] B) {

Set<Integer> uniqueElements = new HashSet<>();

Set<Integer> uniqueset = new HashSet<>();

List<Integer> repeatedElements = new ArrayList<>();

for (int b : B) {

if (!uniqueElements.add(b)) {

if (!repeatedset.contains(b)) {

repeatedElements.add(b);

repeatedset.add(b);

}

if (!repeatedElements.size()==0){

break;

}

}

}

return repeatedElements;}

public static void main(String[] args) {

int[] B = {1, 2, 3, 4, 5, 1, 6, 7, 2, 8, 3, 9, 4};

List<Integer> result = findRepeated(B);

System.out.println("الأعداد المتكررة هي: " + result);

}}

9- 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 GameEntry[] board;

private int n = 0;

public Scoreboard(int capacity) {

board = new GameEntry[capacity];

}

public void add(GameEntry e) throws IllegalStateException {

if (n >= board.length) {

throw new IllegalStateException("لوحة النقاط ممتلئة");

}

board[n] = e;

n++;

}

public GameEntry remove(int i) throws IndexOutOfBoundsException {

if (i < 0 || i >= n) {

throw new IndexOutOfBoundsException("موقع غير صالح");

}

GameEntry temp = board[i];

board[i] = board[n - 1];

board[n - 1] = null;

n--;

return temp;

}}}

10-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.

مولد الارقام العشوائية يعتمد على الصيغة

X{i+1} = (a.Xi + b) mod n

a هو المعامل المضاعف

bهو الثابت الاضافي

nهو المودل

1-تكرار نفس الرقم a=1,b=0

X{i+1} = (Xi )mod 1000

2-تسلسل منتظم بخطوة ثابتة:

a=1,b=10

X{i+1} = (Xi+10 )mod 1000

3-قيمة ثابته لكل الارقام

a=0,b=500

X{i+1} = (0.Xi +500)mod 1000=500

4-دورة قصيرة

b=0,a=250

X{i+1} = (250Xi )mod 1000

11-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?

1. تحليل تأثير الرقم 0:

إذا كان أحد عناصر A يساوي 0، فإن حاصل الضرب X=0.

احتمالية أن يكون عنصر معين في A يساوي 0هي:

P(0) = 1/10

P(no zero) =(9/10)^100

2. حساب احتمالية X=0:

احتمالية أن يحتويA على عنصر واحد على الأقل يساوي 0 هي:

P(x = 0) = 1 - P(no zero)

P(x = 0) = 1 -(9/10)^100

3. حساب قيمة الاحتمالية:

(9/10)^100==.9^100=2.65614\*(10)^-5

P(x = 0) = 1 - 2.65614 (10)^-5= 0.9999744

4. النتيجة النهائية:

الاحتمالية التي يكون فيها X=0 أكبر من 0.99.

الرقم الذي يساوي له Xباحتمالية لا تقل عن 0.99 هو0:

الصيغة النهائية لاحتمالية x = 0:

P(x = 0) = 1 - (9/10)^100

12-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 static void shuffle(int[] A) {

Random rnd = new Random();

for (int i = A.length - 1; i > 0; i--) {

// Swap the current element with a randomly chosen element from the remaining array

int j = rnd.nextInt(i + 1);

int temp = A[i];

A[i] = A[j];

A[j] = temp;

}

}

13-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.

Here's a strategy to track pairs of meeting players and determine the winner(s):

1. Data Structure for Tracking Meetings:

* Bit Array: Employ a 2D boolean array meetings of size n x n. Set meetings[i][j] = true when players i and j meet, indicating a meeting has occurred.
* Alternative: For extremely large n, consider a Bit Set for memory efficiency.

2. Tracking Meetings within meet(i, j):

* When meet(i, j) is called:
  + Set both meetings[i][j] and meetings[j][i] to true (meetings are bidirectional).
  + Increment counters for both players:
    - meetingCount[i]++
    - meetingCount[j]++

3. Determining Winners:

* After each meet(i, j) call:
  + Check if either player i or j has met all other players:
    - If meetingCount[i] == n-1, player i has met everyone and is a winner.
    - Similarly, if meetingCount[j] == n-1, player j is a winner.

4. Handling Ties:

* Maintain a Set<Integer> called winners to track distinct winners.
* Whenever a potential winner is found, add their ID to winners.

5. Code Structure (Example in Java):

Java

// Data structures

boolean[][] meetings = new boolean[n][n];

int[] meetingCount = new int[n];

Set<Integer> winners = new HashSet<>();

void meet(int i, int j) {

meetings[i][j] = meetings[j][i] = true;

meetingCount[i]++;

meetingCount[j]++;

if (meetingCount[i] == n - 1) {

winners.add(i);

// Handle player i's victory (e.g., announce win, mark as finished)

}

if (meetingCount[j] == n - 1) {

winners.add(j);

// Handle player j's victory (e.g., announce win, mark as finished)

}

}

Use code with caution. [Learn more](https://bard.google.com/faq" \l "coding" \t "_blank)

content\_copy

Additional Considerations:

* Game Ending: Decide when the game ends (e.g., when a certain number of winners emerge or after a time limit).
* Winner Announcement: Appropriately announce the winner(s) based on the game's design.
* Memory Optimization: For extremely large n, consider memory-optimized data structures like Bit Sets for storing meeting information.

boolean[][] meetings = new boolean[n][n];

int[] meetingCount = new int[n];

Set<Integer> winners = new HashSet<>();

void meet(int i, int j) {

meetings[i][j] = meetings[j][i] = true;

meetingCount[i]++;

meetingCount[j]++;

if (meetingCount[i] == n - 1) {

winners.add(i);

}

if (meetingCount[j] == n - 1) {

winners.add(j);

}

}

14-Write a Java method that takes two three-dimensional integer arrays and adds them componentwise.

public static int[][] addThreeDimensionalArrays(int[][][] array1, int[][][] array2) {

// Check if the arrays have the same dimensions

if (array1.length != array2.length || array1[0].length != array2[0].length || array1[0][0].length != array2[0][0].length) {

throw new IllegalArgumentException("Arrays must have the same dimensions");

}

// Create a new array to store the result

int[][][] result = new int[array1.length][array1[0].length][array1[0][0].length];

// Add the corresponding elements of the two arrays

for (int i = 0; i < result.length; i++) {

for (int j = 0; j < result[0].length; j++) {

for (int k = 0; k < result[0][0].length; k++) {

result[i][j][k] = array1[i][j][k] + array2[i][j][k];

}

}

}

return result;

}