R-3.1: Generate the next five pseudorandom numbers

The formula for pseudorandom number generation is:
next=(a×current+b)mod n\text{next} = (a \times \text{current} + b) \mod n
nnext=(a×current+b)modn
Using a=12a = 12a=12, b=5b = 5b=5, n=100n = 100n=100, and seed=92\text{seed} = 92seed=92:

1. First: (12×92+5)mod 100=1109mod 100=9(12 \times 92 + 5) \mod 100 = 1109 \mod 100 = 9(12×92+5)mod100=1109mod100=9
2. Second: (12×9+5)mod 100=113mod 100=13(12 \times 9 + 5) \mod 100 = 113 \mod 100 = 13(12×9+5)mod100=113mod100=13
3. Third: (12×13+5)mod 100=161mod 100=61(12 \times 13 + 5) \mod 100 = 161 \mod 100 = 61(12×13+5)mod100=161mod100=61
4. Fourth: (12×61+5)mod 100=737mod 100=37(12 \times 61 + 5) \mod 100 = 737 \mod 100 = 37(12×61+5)mod100=737mod100=37
5. Fifth: (12×37+5)mod 100=449mod 100=49(12 \times 37 + 5) \mod 100 = 449 \mod 100 = 49(12×37+5)mod100=449mod100=49

Answer: The next five numbers are: **9, 13, 61, 37, 49**.

R-3.2: Java method to remove random entries from an array

```
java
Copy code
import java.util.Random;

public class RandomArrayRemoval {
    public static void removeRandomEntries(int[] array) {
        Random random = new Random();
        while (array.length > 0) {
            int index = random.nextInt(array.length);

            // Print removed entry
            System.out.println("Removed entry: " + array[index]);

            // Remove entry at index
            for (int i = index; i < array.length - 1; i++) {
                array[i] = array[i + 1];
            }

            // Resize array
            array = java.util.Arrays.copyOf(array, array.length - 1);
        }
    }

    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        removeRandomEntries(array);
    }
}
```

R-3.3: Caesar cipher for non-English alphabets

Explanation:
To handle non-English alphabets like Greek or Hebrew:

1. Modify the program to accept a custom alphabet string, e.g., "ΑΒΓΔΕ..." for Greek.
2. Replace the modulus %26\% 26%26 (used for English) with %length of the custom alphabet\% \text{length of the custom alphabet}%length of the custom alphabet.
3. Create a map to translate between the custom alphabet and its indices.

R-3.4: Modify `TicTacToe` to throw exception for illegal moves

```
java
Copy code
public class TicTacToe {
    private char[][] board = new char[3][3];
    private char currentPlayer = 'X';
    private boolean gameWon = false;

    public void putMark(int x, int y) {
        if (gameWon) {
            throw new IllegalStateException("Game already won!");
        }
        if (board[x][y] != '\0') {
            throw new IllegalArgumentException("Position already occupied");
        }
        board[x][y] = currentPlayer;
        if (checkWin()) {
            gameWon = true;
        }
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
    }

    private boolean checkWin() {
        // Check rows, columns, and diagonals for a win
        return false; // Simplified
    }
}
```

R-3.13: Difference between shallow and deep equality in arrays

1. **Shallow Equality:** Checks if two arrays reference the same memory location.

   ```
   java
   Copy code
   if (arrayA == arrayB) { ... }  // Shallow equality
   ```

2. **Deep Equality:** Compares the contents of two arrays element-by-element.

   ```
   java
   Copy code
   if (java.util.Arrays.equals(arrayA, arrayB)) { ... }  // Deep equality
   for 1D arrays
   ```

   For 2D arrays:

   ```
   java
   Copy code
   if (java.util.Arrays.deepEquals(arrayA, arrayB)) { ... }  // Deep
   equality for 2D arrays
   ```

R-3.14: Create backup copies of arrays

Three examples to copy an array:

```
java
Copy code
int[] backup1 = original.clone(); // Cloning
int[] backup2 = java.util.Arrays.copyOf(original, original.length); // Using
Arrays.copyOf
int[] backup3 = new int[original.length];
System.arraycopy(original, 0, backup3, 0, original.length); // Using
System.arraycopy
```

C-3.17: Find repeated integer in array

```
java
Copy code
public static int findRepeated(int[] array) {
    Set<Integer> seen = new HashSet<>();
    for (int num : array) {
        if (seen.contains(num)) {
            return num; // Found repeated element
        }
        seen.add(num);
    }
    return -1; // No repetition found
}
```

C-3.18: Find five repeated integers in array

```
java
Copy code
public static List<Integer> findRepeatedFive(int[] array) {
    Set<Integer> seen = new HashSet<>();
    List<Integer> repeated = new ArrayList<>();

    for (int num : array) {
        if (seen.contains(num) && !repeated.contains(num)) {
            repeated.add(num);
            if (repeated.size() == 5) break;
        }
        seen.add(num);
    }
    return repeated;
}
```

C-3.19: Implement `add` and `remove` for Scoreboard

```
java
Copy code
public class Scoreboard {
    private int[] scores = new int[10]; // Assume size 10
    private int size = 0;

    public void add(int score) {
        if (size < scores.length) {
            scores[size++] = score; // Add score without maintaining order
        }
    }

    public void remove(int index) {
        scores[index] = scores[--size]; // Replace removed index with last
element
        scores[size] = 0; // Clear last position
    }
}
```

C-3.22: Shuffle an array

Already provided.

C-3.24: Add two 3D arrays

```
java
Copy code
public static int[][][] addThreeDimensionalArrays(int[][][] array1, int[][][]
array2) {
    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");
    }

    int[][][] result = new
int[array1.length][array1[0].length][array1[0][0].length];
    for (int i = 0; i < array1.length; i++) {
        for (int j = 0; j < array1[i].length; j++) {
            for (int k = 0; k < array1[i][j].length; k++) {
                result[i][j][k] = array1[i][j][k] + array2[i][j][k];
            }
        }
    }
    return result;
}
```