Answer 1)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

import java.util.ArrayList;
import java.util.List;

public class HelloWorld {
    public static int[] findPermutation(String s) {
        int n = s.length();
        int[] perm = new int[n + 1];

        // Initialize the permutation with values 0 to n
        for (int i = 0; i <= n; i++) {
            perm[i] = i;
        }

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

        // Process the string s and find the indices where a new decreasing subsequence starts
        for (int i = 0; i < n; i++) {
            if (s.charAt(i) == 'D') {
                indices.add(i);
            }
        }

        int m = indices.size();
        int[] result = new int[n + 1];

        // Reverse the subarrays between the indices to create a new permutation
        for (int i = 0; i <= m; i++) {
            int start = (i == 0) ? 0 : indices.get(i - 1) + 1;
            int end = (i == m) ? n : indices.get(i);

            for (int j = start, k = end; j <= end; j++, k--) {
                result[j] = perm[k];
            }
        }

        return result;
    }

    public static void main(String[] args) {
        String s = "IDID";
        int[] perm = findPermutation(s);

        System.out.print("Reconstructed permutation: ");
        for (int num : perm) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Answer 2)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

public class HelloWorld {
    public static boolean searchMatrix(int[][] matrix, int target) {
        int m = matrix.length;
        int n = matrix[0].length;

        int left = 0;
        int right = m * n - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int midValue = matrix[mid / n][mid % n];

            if (midValue == target) {
                return true;
            } else if (midValue < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return false;
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 3, 5, 7},
            {10, 11, 16, 20},
            {23, 30, 34, 60}
        };
        int target = 3;

        boolean found = searchMatrix(matrix, target);
        System.out.println("Target found: " + found);
    }
}
```

Answer 3)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

public class HelloWorld {
    public static boolean validMountainArray(int[] arr) {
        int n = arr.length;

        // Check if the array length is less than 3
        if (n < 3) {
            return false;
        }

        int i = 0;

        // Find the peak of the mountain
        while (i < n - 1 && arr[i] < arr[i + 1]) {
            i++;
        }

        // Check if the peak is at the beginning or end
        if (i == 0 || i == n - 1) {
            return false;
        }

        // Check the decreasing part of the mountain
        while (i < n - 1 && arr[i] > arr[i + 1]) {
            i++;
        }

        return i == n - 1;
    }
    public static void main(String[] args) {
        int[] arr = {2, 1};

        boolean isValidMountain = validMountainArray(arr);
        System.out.println("Is valid mountain array: " + isValidMountain);
    }
}
```

Answer 4)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

import java.util.HashMap;
import java.util.Map;

public class HelloWorld {
    public static int findMaxLength(int[] nums) {
        int maxLength = 0;
        int count = 0;
        Map<Integer, Integer> countMap = new HashMap<>();
        countMap.put(0, -1);

        for (int i = 0; i < nums.length; i++) {
            count += nums[i] == 1 ? 1 : -1;

            if (countMap.containsKey(count)) {
                maxLength = Math.max(maxLength, i - countMap.get(count));
            } else {
                countMap.put(count, i);
            }
        }

        return maxLength;
    }

    public static void main(String[] args) {
        int[] nums = {0, 1};

        int maxLength = findMaxLength(nums);
        System.out.println("Maximum length of contiguous subarray: " + maxLength);
    }
}
```

Answer 5)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

import java.util.Arrays;

public class HelloWorld {
    public static int minProductSum(int[] nums1, int[] nums2) {
        Arrays.sort(nums1);
        Arrays.sort(nums2);

        int sum = 0;
        int n = nums1.length;

        for (int i = 0; i < n; i++) {
            sum += nums1[i] * nums2[n - i - 1];
        }

        return sum;
    }

    public static void main(String[] args) {
        int[] nums1 = {5, 3, 4, 2};
        int[] nums2 = {4, 2, 2, 5};

        int minProductSum = minProductSum(nums1, nums2);
        System.out.println("Minimum product sum: " + minProductSum);
    }
}
```

Answer 6)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class HelloWorld {
    public static int[] findOriginalArray(int[] changed) {
        int n = changed.length;

        // If the number of elements is odd, it cannot be a doubled array
        if (n % 2 != 0) {
            return new int[0];
        }

        // Count the frequency of each element in the changed array
        Map<Integer, Integer> countMap = new HashMap<>();
        for (int num : changed) {
            countMap.put(num, countMap.getOrDefault(num, 0) + 1);
        }

        // Sort the array in ascending order
        Arrays.sort(changed);

        int[] original = new int[n / 2];
        int index = 0;

        // Iterate over the changed array
        for (int num : changed) {
            // Check if the current number is twice the value of another number
            if (countMap.getOrDefault(num, 0) > 0 && countMap.getOrDefault(num * 2, 0) > 0) {
                original[index] = num;
                index++;

                // Decrement the counts of the current number and its doubled value
                countMap.put(num, countMap.get(num) - 1);
                countMap.put(num * 2, countMap.get(num * 2) - 1);
            }
        }

        // If all elements have been paired and used, return the original array
        if (index == n / 2) {
            return original;
        } else {
            return new int[0]; // Return an empty array if not a doubled array
        }
    }
    public static void main(String[] args) {
        int[] changed = {1, 3, 4, 2, 6, 8};

        int[] original = findOriginalArray(changed);

        System.out.println("Original array: " + Arrays.toString(original));
    }
}
```

Answer 7)

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

public class HelloWorld {
    public static int[][] generateMatrix(int n) {
        int[][] matrix = new int[n][n];

        int num = 1; // Starting number
        int rowStart = 0;
        int rowEnd = n - 1;
        int colStart = 0;
        int colEnd = n - 1;

        while (rowStart <= rowEnd && colStart <= colEnd) {
            // Fill the top row
            for (int col = colStart; col <= colEnd; col++) {
                matrix[rowStart][col] = num++;
            }
            rowStart++;

            // Fill the right column
            for (int row = rowStart; row <= rowEnd; row++) {
                matrix[row][colEnd] = num++;
            }
            colEnd--;

            // Fill the bottom row
            if (rowStart <= rowEnd) {
                for (int col = colEnd; col >= colStart; col--) {
                    matrix[rowEnd][col] = num++;
                }
                rowEnd--;
            }

            // Fill the left column
            if (colStart <= colEnd) {
                for (int row = rowEnd; row >= rowStart; row--) {
                    matrix[row][colStart] = num++;
                }
                colStart++;
            }
        }

        return matrix;
    }

    public static void main(String[] args) {
        int n = 3;
        int[][] matrix = generateMatrix(n);

        // Print the generated matrix
        for (int[] row : matrix) {
            System.out.println(Arrays.toString(row));
        }
    }
}
```

Answer 8)