Q - 1 Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2),..., (an, bn) such that the sum of min(ai, bi) for all i is maximized. Return the maximized sum.

ANS

```
import java.util.Arrays;

import java.util.Scanner;


public class ArrayPairSum {

  public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);


    System.out.print("Enter the number of elements (n): ");

    int n = sc.nextInt();

    int[] nums = new int[2 * n];


    System.out.println("Enter the elements:");

    for (int i = 0; i < 2 * n; i++) {

      nums[i] = sc.nextInt();

    }



    Arrays.sort(nums);



    int sum = 0;

    for (int i = 0; i < 2 * n; i += 2) {

      sum += nums[i];

    }
```

```
        System.out.println("Maximized sum: " + sum);

        sc.close();
    }
}
```

Q 2- Alice has n candies, where the ith candy is of type candyType[i]. Alice noticed that she started to gain weight, so she visited a doctor.

The doctor advised Alice to only eat n / 2 of the candies she has (n is always even). Alice likes her candies very much, and she wants to eat the maximum number of different types of candies while still following the doctor's advice.

Given the integer array candyType of length n, return the maximum number of different types of candies she can eat if she only eats n / 2 of them.

ANS -

```
import java.util.Scanner;
import java.util.sc;

public class MaxDifferentCandies {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of candies (n): ");
        int n = sc.nextInt();
        int[] candyType = new int[n];

        System.out.println("Enter the candy types:");
        for (int i = 0; i < n; i++) {
            candyType[i] = sc.nextInt();
        }

        int maxTypes = maxDifferentCandies(candyType);

        System.out.println("Maximum number of different types of candies Alice can eat: " + maxTypes);

        sc.close();
    }

    public static int maxDifferentCandies(int[] candyType) {
        int maxTypes = 0;
        int[] types = new int[200001];
        for (int type : candyType) {
            if (types[type] == 0) {
                maxTypes++;
                types[type] = 1;
            }
        }

        return Math.min(maxTypes, candyType.length / 2);
    }
}
```

Q 3 - We define a harmonious array as an array where the difference between its maximum value and its minimum value is exactly 1.

Given an integer array nums, return the length of its longest harmonious subsequence among all its possible subsequences.

A subsequence of an array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements.

ANS -- import java.util.Arrays;

```
import java.util.Scanner;

public class LoHaSub {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] nums = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        int LngSub = findLongest(nums);

        System.out.println("Length of the longest harmonious subsequence: " + LngSub);

        scanner.close();
    }

    public static int findLongest(int[] nums) {
        Arrays.sort(nums);

        int LngSub = 0;
        int start = 0;
        for (int end = 0; end < nums.length; end++) {
            while (nums[end] - nums[start] > 1) {
                start++;
            }

            if (nums[end] - nums[start] == 1) {
                LngSub = Math.max(LngSub, end - start + 1);
            }
        }

        return LngSub;
    }
}
```

Q 4 You have a long flowerbed in which some of the plots are planted, and some are not.

However, flowers cannot be planted in adjacent plots.

Given an integer array flowerbed containing 0's and 1's, where 0 means empty and 1 means not empty, and an integer n, return true if n new flowers can be planted in the flowerbed without violating the no-adjacent-flowers rule and false otherwise.

ANS --

```
import java.util.Scanner;

public class Flower{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements in the flowBed: ");
        int n = scanner.nextInt();
        int[] flowBed = new int[n];
        System.out.println("Enter the elements of the flowBed (0 or 1):");
        for (int i = 0; i < n; i++) {
            flowBed[i] = scanner.nextInt();
        }

        System.out.print("Enter the number of new flowers to be planted: ");
        int NewFl = scanner.nextInt();

        boolean flower = flower(flowBed, NewFl);

        System.out.println("Can the flowers be planted without violating the rule? " + flower);

        scanner.close();
    }

    public static boolean flower(int[] flowBed, int n) {
        int count = 0;
        int i = 0;

        while (i < flowBed.length) {
            if (flowBed[i] == 0 && (i == 0 || flowBed[i - 1] == 0) && (i == flowBed.length - 1 || flowBed[i +
1] == 0)) {
                flowBed[i] = 1;
                count++;
            }

            if (count >= n) {
                return true;
            }

            i++;
        }

        return false;
    }
}
```

Question 5

Given an integer array nums, find three numbers whose product is maximum and return the maximum product.

Example 1:

Input: nums = [1,2,3]

Output: 6

ANS - import java.util.Scanner;

```
public class MaximumProductOfThreeNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] nums = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        int maxProduct = maximumProduct(nums);
        System.out.println("Maximum product of three numbers: " + maxProduct);

        scanner.close();
    }

    public static int maximumProduct(int[] nums) {
        int max1 = Integer.MIN_VALUE;
        int max2 = Integer.MIN_VALUE;
        int max3 = Integer.MIN_VALUE;
        int min1 = Integer.MAX_VALUE;
        int min2 = Integer.MAX_VALUE;

        for (int num : nums) {
            if (num > max1) {
                max3 = max2;
                max2 = max1;
                max1 = num;
            } else if (num > max2) {
                max3 = max2;
                max2 = num;
            } else if (num > max3) {
                max3 = num;
            }

            if (num < min1) {
                min2 = min1;
                min1 = num;
            } else if (num < min2) {
                min2 = num;
            }
        }

        int product1 = max1 * max2 * max3;
        int product2 = max1 * min1 * min2;

        return Math.max(product1, product2);
    }
}
```

Question 6

Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1.

You must write an algorithm with O(log n) runtime complexity.

ANS import java.util.Scanner;

```
public class BinarySearch {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] nums = new int[n];

        System.out.println("Enter the elements of the array in ascending order:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        System.out.print("Enter the target value: ");
        int target = scanner.nextInt();

        int index = search(nums, target);

        System.out.println("Index of the target value: " + index);

        scanner.close();
    }

    public static int search(int[] nums, int target) {
        int left = 0;
        int right = nums.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return -1;
    }
}
```

Q An array is monotonic if it is either monotone increasing or monotone decreasing.

An array nums is monotone increasing if for all i <= j, nums[i] <= nums[j]. An array nums is monotone decreasing if for all i <= j, nums[i] >= nums[j].

Given an integer array nums, return true if the given array is monotonic, or false otherwise.

ANS - import java.util.Scanner;

```
public class MonotonicArray {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the number of elements in the array: ");
    int n = scanner.nextInt();
    int[] nums = new int[n];

    System.out.println("Enter the elements of the array:");
    for (int i = 0; i < n; i++) {
      nums[i] = scanner.nextInt();
    }

    boolean isMonotonic = isMonotonic(nums);

    System.out.println("Is the array monotonic? " + isMonotonic);

    scanner.close();
  }

  public static boolean isMonotonic(int[] nums) {
    boolean increasing = true;
```

```
        boolean decreasing = true;

        for (int i = 1; i < nums.length; i++) {
            if (nums[i] < nums[i - 1]) {
                increasing = false;
            }
            if (nums[i] > nums[i - 1]) {
                decreasing = false;
            }
        }

        return increasing || decreasing;
    }
}
```

Question 8

You are given an integer array nums and an integer k.

In one operation, you can choose any index i where 0 <= i < nums.length and change nums[i] to nums[i] + x where x is an integer from the range [-k, k]. You can apply this operation at most once for each index i.

The score of nums is the difference between the maximum and minimum elements in nums.

Return the minimum score of nums after applying the mentioned operation at most once for each index in it.

Explanation: The score is max(nums) - min(nums) = 1 - 1 = 0.

ANS -- import java.util.Arrays;

```
import java.util.Scanner;
```

```
public class MinimumScore {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] nums = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        System.out.print("Enter the value of k: ");
        int k = scanner.nextInt();

        int minimumScore = minScore(nums, k);

        System.out.println("Minimum score: " + minimumScore);

        scanner.close();
    }

    public static int minScore(int[] nums, int k) {
        int n = nums.length;
        int minScore;

        if (n == 1) {
            return 0;
        }
        Arrays.sort(nums);


        int maxWithoutOp = nums[n - 1] - nums[0];
        int maxWithOp = Math.max(nums[n - 1] - k, nums[0] + k);


        minScore = Math.min(maxWithoutOp, maxWithOp);


        return minScore;
    }
}
```