**Question 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.**

**Example 1: Input: nums = [1,4,3,2]**

**Output: 4**

**Prgm:**

class Solution {

public int arrayPairSum(int[] nums) {

Arrays.sort(nums);

int len = nums.length;

int sum = 0;

for (int i = 0; i < len - 1; i += 2) {

sum += nums[i];

}

return sum;

}

}

**Question 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.**

**Example 1:**

**Input: nums = [1,3,2,2,5,2,3,7]**

**Output: 5**

**Prgm:**

import java.util.Arrays;

class LongestHarmoniousSubsequence {

public static int findLHS(int[] nums) {

int maxLen = 0;

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

int count = 0;

boolean found = false;

for (int j = 0; j < nums.length; j++) {

if (nums[j] == nums[i] || nums[j] == nums[i] + 1) {

count++;

}

if (nums[j] == nums[i] + 1) {

found = true;

}

}

if (found) {

maxLen = Math.max(maxLen, count);

}

}

return maxLen;

}

public static void main(String[] args) {

int[] nums1 = {1,3,2,2,5,2,3,7};

System.out.println("Input: nums1 = " + Arrays.toString(nums1));

System.out.println("Output: " + findLHS(nums1));

int[] nums2 = {1,2,3,4};

System.out.println("Input: nums2 = " + Arrays.toString(nums2));

System.out.println("Output: " + findLHS(nums2));

}

}

**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**

**Prgm:**

class maximumProducts {

public int maximumProduct(int[] nums) {

Arrays.sort(nums);

return Math.max(nums[0] \* nums[1] \* nums[nums.length - 1], nums[nums.length - 1] \* nums[nums.length - 2] \* nums[nums.length - 3]);

}

public static void main(String[] args) {

int nums[] = {1,2,3};

maximumProducts mp = new maximumProducts();

System.out.println(mp.maximumProduct(nums));

}

}

**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.**

**Input: nums = [-1,0,3,5,9,12], target = 9**

**Output: 4**

**PRGM:**

class BinarySearch {

public int search(int[] nums, int target) {

int left = 0, right = nums.length;

while (left < right) {

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

if (nums[mid] <= target) {

left = mid + 1;

} else {

right = mid;

}

}

if (left > 0 && nums[left - 1] == target) {

return left - 1;

} else {

return -1;

}

}

public static void main(String args[])

{

BinarySearch ob = new BinarySearch();

int nums[] = { -1,0,3,5,9,12 };

int n = nums.length;

int x = 9;

int result = ob.search(nums, x);

if (result == -1)

System.out.println(

"Target is not exists in nums");

else

System.out.println("Element is present at "

+ "index " + result);

}

}

**Question 7**

**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.**

**Example 1:**

**Input: nums = [1,2,2,3]**

**Output: true**

**Prgm:**

class Monotonic{

public static boolean isMonotonic(int[] A) {

boolean isincr = true;

boolean isdec = true;

int n=A.length;

for (int i = 0; i < n- 1; ++i) {

if (A[i] > A[i+1])

isincr = false;

if (A[i] < A[i+1])

isdec = false;

}

return isincr || isdec;

}

public static void main(String[] args) {

int [] arr = {1,2,2,3};

boolean ans= isMonotonic(arr);

System.out.println(ans);

}

}

**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.**

**Example 1:**

**Input: nums = [1], k = 0**

**Output: 0**

**Prgm:**

class SmallestRangeI {

public static int smallestRangeI(int[] A, int K) {

int mx = A[0], mn = A[0];

for (int a : A) {

mx = Math.max(mx, a);

mn = Math.min(mn, a);

}

return Math.max(0, mx - mn - 2 \* K);

}

public static void main(String[] args){

int A[] = {1};

int k = 0;

System.out.println(smallestRangeI(A,k));

}

}