# Q1. Given an array of N integers. Your task is to print the sum of all of the integers.

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 07/05/2023  
 \*/*public class QuestionOneSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*sumOfAllIntegers*(4, new int[]{1, 2, 3, 4}));  
 System.*out*.println(*sumOfAllIntegers*(6, new int[]{5, 8, 3, 10, 22, 45}));  
 System.*out*.println(*sumOfAllIntegers*(8, new int[]{}));  
 }  
  
 public static int sumOfAllIntegers(int n, int[] array) {  
 int sum = 0;  
 if (array == null || array.length == 0) return sum;  
 for (int i = 0; i < n; i++) {  
 sum += array[i];  
 }  
 return sum;  
 }  
}

# Q2. Given an array A[] of N integers and an index Key. Your task is to print the element present at index key in the array.

## Solution*:*

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 07/05/2023  
 \*/*public class QuestionTwoSolution {  
 *//main method* public static void main(String[] args) {  
 System.*out*.println(*findElementAtIndex*(5, 2, new int[]{10, 20, 30, 40, 50}));  
 System.*out*.println(*findElementAtIndex*(7, 4, new int[]{10, 20, 30, 40, 50, 60, 70}));  
 System.*out*.println(*findElementAtIndex*(8, 9, new int[]{}));  
 }  
  
 *//method to find element at Kth position* public static int findElementAtIndex(int N, int key, int[] A) {  
 int element = 0;  
 if (A == null || A.length == 0) return element;  
 for (int i = 0; i < N; i++) {  
 if (A[key] == A[i])  
 element = A[i];  
 }  
 return element;  
 }  
}

# Q3. Given a sorted array A of size N. Find number of elements which are less than or equal to given element X.

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 07/05/2023  
 \*/*public class QuestionThreeSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findNumberOfElementsLessThanOrEqual*(  
 6, new int[]{1, 2, 4, 5, 8, 10}, 9)  
 );  
 System.*out*.println(*findNumberOfElementsLessThanOrEqual*(  
 7, new int[]{1, 2, 2, 2, 5, 7,9}, 2)  
 );  
 }  
  
 public static int findNumberOfElementsLessThanOrEqual(int N, int[] A, int X) {  
 if (A == null || A.length == 0) return 0;  
 int count = 0;  
 for (int i = 0; i < N; i++) {  
 if (A[i] <= X) count++;  
 }  
 return count;  
 }  
}

# Q4. You are given an array A of size N. You need to print elements of A in alternate order (starting from index 0).

## Solution:

import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/05/2023  
 \*/*public class QuestionFourSolution {  
 public static void main(String[] args) {  
 *printElementsInAlternativeOrderMethod1*(4, new int[]{1, 2, 3, 4});  
 System.*out*.println();  
 *printElementsInAlternativeOrderMethod2*(5, new int[]{1, 2, 3, 4, 5});  
 }  
  
 *//method 1: print the alternative number directly by traversing the array* private static void printElementsInAlternativeOrderMethod1(int N, int[] A) {  
 for (int i = 0; i < N; i = i + 2) {  
 System.*out*.print(A[i] + " ");  
 }  
 }  
  
 *//method 2: Traverse the array, store alternative number in a array list or array then print it* private static void printElementsInAlternativeOrderMethod2(int N, int[] A) {  
 List<Integer> result = new ArrayList<>();  
 for (int i = 0; i < N; i = i + 2) {  
 result.add(A[i]);  
 }  
 result.forEach((element) -> System.*out*.print(element + " "));  
 }  
}

# Q5. Given an array Arr of N positive integers. Your task is to find the elements whose value is equal to that of its index value (Consider 1-based indexing)

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/05/2023  
 \*/*public class QuestionFiveSolution {  
 public static void main(String[] args) {  
 *findElementsEqualToIndex*(5, new int[]{15, 2, 45, 12, 7});  
 System.*out*.println();  
 *findElementsEqualToIndex*(1, new int[]{1});  
 System.*out*.println();  
 *findElementsEqualToIndex*(8, new int[]{45, 78, 3, 4, 5, 41, 89, 8});  
 }  
  
 public static void findElementsEqualToIndex(int N, int[] Arr) {  
 for (int i = 0; i < N; i++) {  
 if (i + 1 == Arr[i]) System.*out*.print(Arr[i]);  
 }  
 }  
}

# Q6. Given an array of size N and you have to tell whether the array is perfect or not. An array is said to be perfect if it's reverse array matches the original array. If the array is perfect then print "PERFECT" else print "NOT PERFECT.

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/05/2023  
 \*/*public class QuestionSixSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*decidePerfectArray*(new int[]{1, 2, 3, 2, 1}));  
 System.*out*.println(*decidePerfectArray*(new int[]{1, 2, 3, 4, 5}));  
 System.*out*.println(*decidePerfectArray*(new int[]{7, 8, 9, 10, 11, 10, 9, 8, 7}));  
 }  
  
 public static String decidePerfectArray(int[] Arr) {  
 if (Arr == null || Arr.length == 0) return null;  
 int length = Arr.length;  
 int[] reverseArray = new int[length];  
 int j = 0;  
 for (int i = length - 1; i >= 0; i--) {  
 reverseArray[j] = Arr[i];  
 j++;  
 }  
 for (int i = 0; i < length; i++) {  
 if (Arr[i] != reverseArray[i]) return "NOT PERFECT";  
 }  
 return "PERFECT";  
 }  
}

# Q7. Given an array of length **N**, at each step it is reduced by 1 element. In the first step the maximum element would be removed, while in the second step minimum element of the remaining array would be removed, in the third step again the maximum and so on. Continue this till the array contains only 1 element. And find the final element remaining in the array.

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/05/2023  
 \*/*public class QuestionSevenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findLeftElementMethod1*(7, new int[]{7, 8, 3, 4, 2, 9, 5}));  
 System.*out*.println(*findLeftElementMethod2*(8, new int[]{8, 1, 2, 9, 4, 3, 7, 5}));  
 System.*out*.println(*findLeftElementMethod1*(7, new int[]{9, 11, 1, 4, 8, 6, 1}));  
 System.*out*.println(*findLeftElementMethod1*(6, new int[]{1, 2, 3, 3, 4, 5}));  
 System.*out*.println(*findLeftElementMethod1*(6, new int[]{1, 2, 3, 4, 5, 5}));  
 }  
  
 *//Sort using bubble sort method, then find the last remaining element* public static int findLeftElementMethod1(int N, int[] A) {  
 for (int i = 0; i <= N - 1; i++) {  
 for (int j = i; j <= N - 1; j++) {  
 if (A[i] > A[j]) {  
 int temp = A[i];  
 A[i] = A[j];  
 A[j] = temp;  
 }  
 }  
 }  
 if (N % 2 == 0) return A[(N / 2) - 1];  
 return A[N / 2];  
 }  
  
 *//sort using Arrays.sort() method* public static int findLeftElementMethod2(int N, int[] A) {  
 Arrays.*sort*(A);  
 if (N % 2 == 0) return A[(N / 2) - 1];  
 return A[N / 2];  
 }  
}

# Q8. Given an array of **N** distinct elements, the task is to find all elements in array except two greatest elements in sorted order.

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/05/2023  
 \*/*public class QuestionEightSolution {  
 public static void main(String[] args) {  
 *findElementsExceptTwoGreatest*(new int[]{2, 8, 7, 1, 5});  
 System.*out*.println();  
 *findElementsExceptTwoGreatest*(new int[]{7, -2, 3, 4, 9, -1});  
 }  
  
 public static void findElementsExceptTwoGreatest(int[] a) {  
 int len = a.length;  
 *sort*(len, a);  
 for (int i = 0; i < len - 2; i++) {  
 System.*out*.print(a[i] + " ");  
 }  
 }  
  
 *//method to sort the array* public static void sort(int len, int[] arr) {  
 for (int i = 0; i <= len - 1; i++) {  
 for (int j = i; j <= len - 1; j++) {  
 if (arr[i] > arr[j]) {  
 int temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
 }  
 }  
 }  
 }  
}

# Q9. Write a program to find the sum of the given series 1+2+3+ . . . . . .(**N** terms)

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/05/2023  
 \*/*public class QuestionNineSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findSumOfSeries*(5));  
 System.*out*.println(*findSumOfSeries*(10));  
 System.*out*.println(*findSumOfSeries*(1));  
 }  
  
 public static int findSumOfSeries(int N) {  
 int sum = 0;  
 for (int i = 1; i <= N; i++) {  
 sum += i;  
 }  
 return sum;  
 }  
}

# Q10. Given a number **N**. Your task is to check whether it is fascinating or not.

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/05/2023  
 \*/*public class QuestionTenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*checkFascinatingNumber*(192));  
 System.*out*.println(*checkFascinatingNumber*(853));  
 }  
  
 public static String checkFascinatingNumber(int N) {  
 int multiplyResult1 = N \* 2;  
 int multiplyResult2 = N \* 3;  
 String concatenatedResult = String.*valueOf*(N) + multiplyResult1 + multiplyResult2;  
 for (int i = 1; i <= 9; i++) {  
 if (!concatenatedResult.contains(String.*valueOf*(i))) return "Not Fascinating";  
 }  
 return "Fascinating";  
 }  
}

# Bonus Question: Given an array of even size **N**, task is to find minimum value that can be added to an element so that array become balanced. An array is balanced if the sum of the left half of the array elements is equal to the sum of right half.

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 08/05/2023  
 \*/*public class BonusQuestionSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findBalanceValue*(4, new int[]{1, 5, 3, 2}));  
 System.*out*.println(*findBalanceValue*(6, new int[]{1, 2, 1, 2, 1, 3}));  
 }  
  
 public static int findBalanceValue(int N, int[] arr) {  
 int leftSum = 0;  
 int rightSum = 0;  
 for (int i = 0; i < N / 2; i++) {  
 leftSum += arr[i];  
 }  
 for (int i = N / 2; i < N; i++) {  
 rightSum += arr[i];  
 }  
 if (leftSum > rightSum) return leftSum - rightSum;  
 return rightSum - leftSum;  
 }  
}