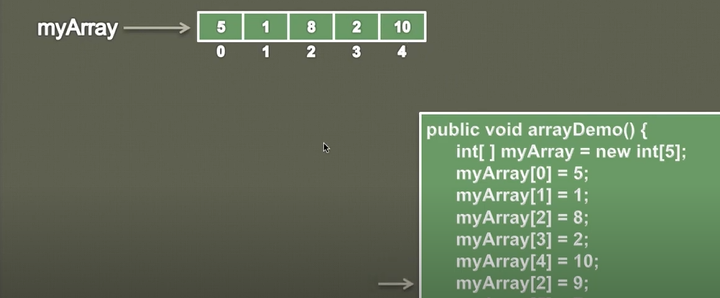
# Lab Task(s):

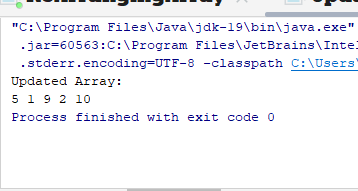
**Exercise 1**

1. Adding and updating an element in one dimensional array.



package FirstLabTasks;  
  
public class UpdateArray {  
  
 public static void PrintArray(int[]myArray){  
 for(int i=0; i<myArray.length; i++){  
 System.*out*.print(myArray[i]+" ");  
  
 }  
  
 }  
  
 public static void main(String[] args) {  
  
 int myArray[]=new int[5];  
 myArray[0]=5;  
 myArray[1]=1;  
 myArray[2]=8;  
 myArray[3]=2;  
 myArray[4]=10;  
 myArray[2]=9;  
  
 System.*out*.println("Updated Array:");  
 *PrintArray*(myArray);  
  
  
  
 }  
}

**OUTPUT**



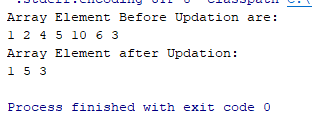
1. In this problem, you have to implement the **int [] removeEven(int[] arr)** method, which removes all the even elements from the array and returns back updated array.



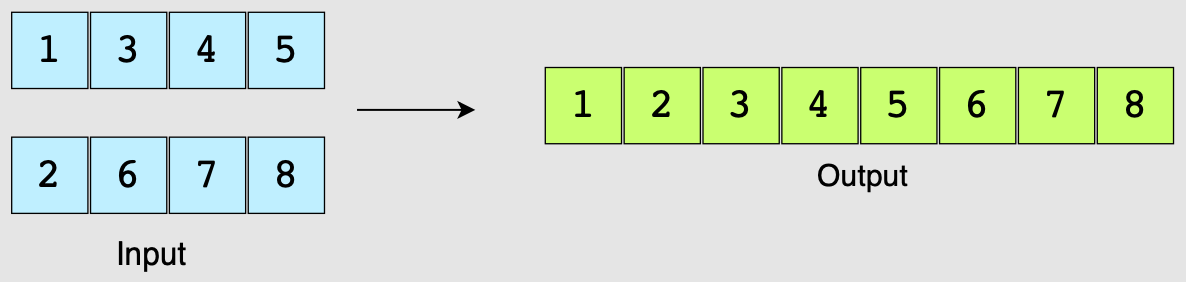
package FirstLabTasks;  
  
public class OddArrayElement {  
  
 public static void PrintArray(int[] Array) {  
 for(int i = 0; i < Array.length; ++i) {  
 System.*out*.print(Array[i] + " ");  
 }  
  
 System.*out*.println();  
 }  
  
 public static int[] removeEven(int[] Array) {  
  
 int Odd = 0;  
 for(int i = 0; i < Array.length; ++i) {  
 if (Array[i] % 2 != 0) {  
 ++Odd;  
 }  
 }  
  
 int[] Result = new int[Odd];  
 int index = 0;  
  
 for(int i = 0; i < Array.length; i++) {  
 if (Array[i] % 2 != 0) {  
 Result[index] = Array[i];  
 index++;  
 }  
 }  
  
 return Result;  
 }  
  
 public static void main(String[] args) {  
 int[] Array = new int[]{1, 2, 4, 5, 10, 6, 3};  
 System.*out*.println("Array Element Before Updation are:");  
 *PrintArray*(Array);  
 int[] result = *removeEven*(Array);

System.*out*.println("Array Element after Updation: ");  
 *PrintArray*(result);  
 }  
}

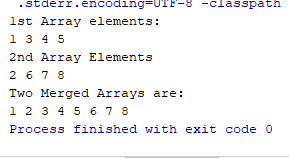
**OUTPUT**



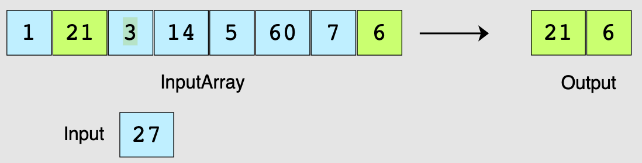
3:In this problem, given two sorted arrays, you have to implement the **int[] mergeArrays(int[] arr1, int[] arr2)** method, which returns an array consisting of all elements of both arrays in a sorted way.



package FirstLabTasks;  
  
import java.util.Arrays;  
  
public class TwoArraySorting {  
public static void PrintArray(int[]arr1,int[] arr2){  
 System.*out*.println("1st Array elements:");  
 for(int i=0; i<arr1.length; i++){  
 System.*out*.print(arr1[i]+" ");  
  
 }  
 System.*out*.println();  
  
 System.*out*.println("2nd Array Elements");  
  
 for (int i=0;i<arr2.length; i++){  
 System.*out*.print(arr2[i]+" ");  
 }  
 System.*out*.println();  
  
  
}  
  
public static int[] mergeArrays(int[] arr1, int[] arr2) {  
  
 int length1 = arr1.length;  
 int length2 = arr2.length;  
  
 int Result[] = new int[length1 + length2];  
 for (int i = 0; i < length1; i++) {  
 Result[i] = arr1[i];  
  
 }  
  
 for (int i = 0; i < length2; i++) {  
 Result[length1 + i] = arr2[i];  
  
 }  
  
 return Result;  
  
}  
  
 public static void main(String[] args) {  
 int arr1[]=new int[]{1,3,4,5};  
 int arr2[]=new int[]{2,6,7,8};  
  
 *PrintArray*(arr1, arr2);  
 int[] result = *mergeArrays*(arr1, arr2);  
  
 *mergeArrays*(arr1, arr2);  
  
 Arrays.*sort*(result);  
 System.*out*.println("Two Merged Arrays are:");  
  
 for(int i = 0; i < result.length; ++i) {  
 System.*out*.print(result[i] + " ");  
 }  
  
 }  
}



4: In this problem, you have to implement the **int[] findSum(int[] arr, int n)** method, which will take a number **n**, and an array **arr** as input and returns an **array of two integers** that add up to n in an array. You are required to return only one such pair. If no such pair is found then simply return the array.



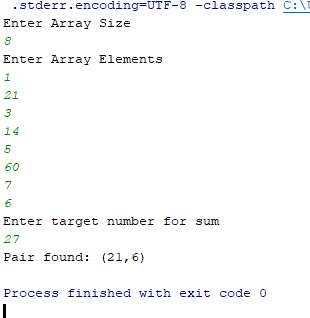
package FirstLabTasks;  
  
 import java.util.Scanner;  
  
 public class ArrayTask4 {  
  
 public static int[] findSum(int[] arr, int n) {

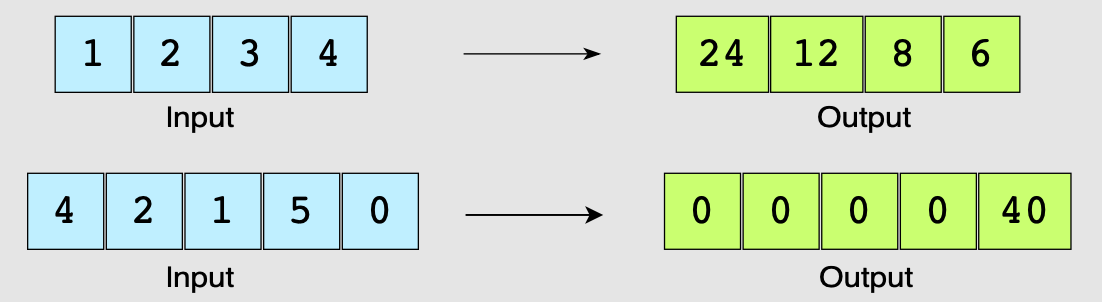
for(int i = 0; i < arr.length - 1; ++i) {  
 for(int j = i + 1; j < arr.length; ++j) {  
 if (arr[i] + arr[j] == n) {  
 int[] pair = new int[]{arr[i], arr[j]};  
 return pair;  
 }  
 }  
 }

return arr;  
 }

public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Enter Array Size");  
 int size = sc.nextInt();  
 int[] arr = new int[size];  
 System.*out*.println("Enter Array Elements");  
  
 int n;  
 for(n = 0; n < size; ++n) {  
 arr[n] = sc.nextInt();  
 }  
  
 System.*out*.println("Enter target number for sum");  
 n = sc.nextInt();  
 *findSum*(arr, n);  
 int[] result = *findSum*(arr, n);  
 if (result.length == 2) {  
 System.*out*.println("Pair found: (" + result[0] + "," + result[1] + ")");  
 } else {  
 System.*out*.println("Pair not found");  
 }  
  
 }

**OUTPUT**

} 

**5:**In this problem, you have to implement the **int[] findProduct(int[] arr)** method which will modify arr in such a way that in the output, each index i will contain the product of all elements present in arr except the element stored on that index i.

package FirstLabTasks;  
  
import java.util.Scanner;  
  
public class ExceptSelfProduct {  
  
 public static int[] findProduct(int[] arr){  
 int temp=1;  
 int result[]=new int[arr.length];  
  
 for(int i=0; i<arr.length;i++){ *//for left side product* result[i]=temp;  
 temp=temp\*arr[i];  
  
 }  
  
 temp=1;  
 for(int i= arr.length-1;i>=0;i--){  
 *//for right side product of array* result[i]=result[i]\*temp;  
 result[i] = result[i] \* temp;  
 temp = temp \* arr[i];  
  
  
 }  
  
 return result;  
  
 }  
  
  
 public static void main(String[] args) {  
  
 Scanner sc=new Scanner(System.in);  
 System.out.println("Enter size of Array");  
 int size=sc.nextInt();

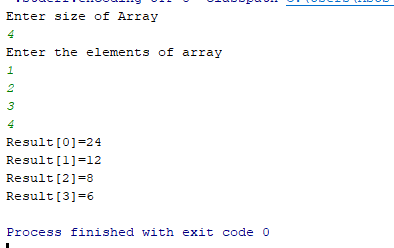
int arr[]=new int[size];

System.*out*.println("Enter the elements of array");

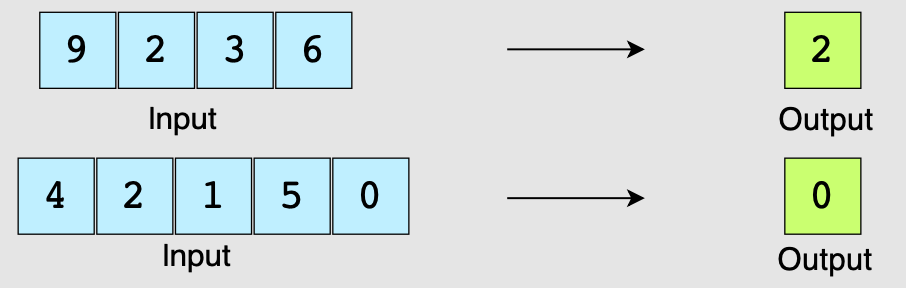
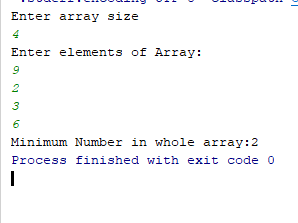
for (int i=0; i<arr.length; i++){  
 arr[i]=sc.nextInt();  
 }

int Result[]=findProduct(arr);  
 for(int i=0; i< Result.length;i++){  
 System.out.println("Result["+i+"]="+Result[i]+" ");  
 }  
 }  
}

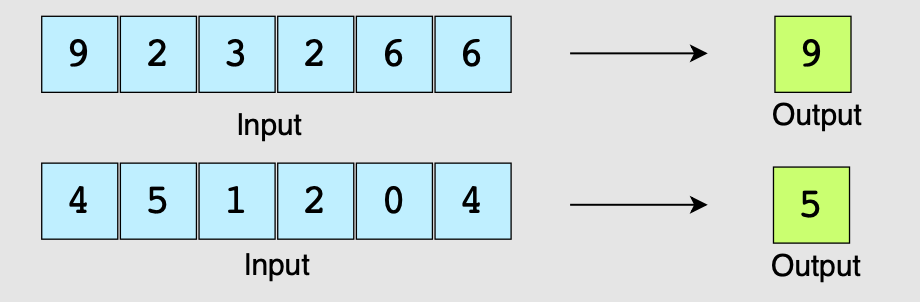
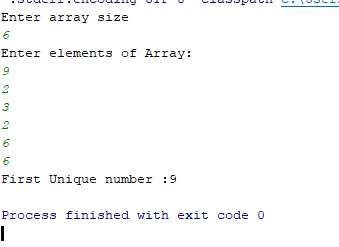
**OUTPUT**



6: In this problem, you have to implement the **int findMinimum(int[] arr)** method, which will traverse the whole array and find the smallest number in the array.

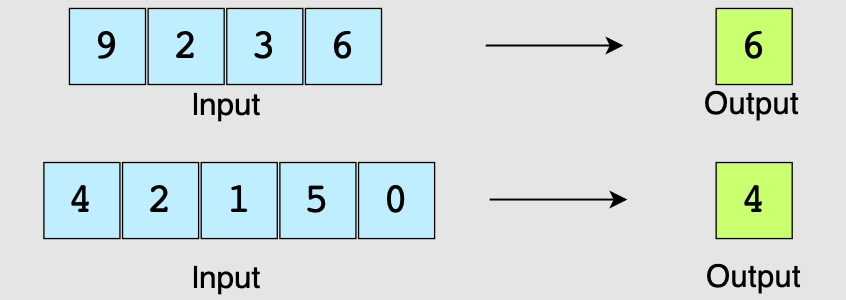
package FirstLabTasks;  
import java.util.Scanner;  
public class MinimumNumber {  
  
 public static int findMinimum(int[] arr){  
  
 int minimum=arr[0];  
 for(int i=0; i<arr.length; i++){  
 if(arr[i]<minimum){  
 minimum=arr[i];  
 }  
 }  
 return minimum;  
 }  
  
 public static void main(String[] args) {  
 Scanner sc=new Scanner(System.*in*);  
 System.*out*.println("Enter array size");  
 int size=sc.nextInt();  
  
 int arr[]=new int[size];  
 System.*out*.println("Enter elements of Array:");  
 for(int i=0; i<size; i++){  
 arr[i]=sc.nextInt();  
 }  
 int Minimum\_number=*findMinimum*(arr);  
 System.*out*.print("Minimum Number in whole array:"+Minimum\_number);  
  
 }  
}  
 **OUTPUT**

7: In this problem, you have to implement the **int findFirstUnique(int[] arr)** method that will look for a first unique integer, which appears only once in the whole array. The function returns -1 if no unique number is found.

package FirstLabTasks;  
  
import java.util.Scanner;  
  
public class UniqueElement {  
  
 public static int findFirstUnique(int[] arr) {  
  
  
 for (int i = 0; i <arr.length; i++) {  
 int count=0;  
  
 for(int j=0;j<arr.length;j++){  
 if(i!=j && arr[i]==arr[j]){  
 count++;  
 break;  
 }  
  
 }  
 if(count==0){  
 return arr[i];  
 }  
  
 }  
  
 return -1; *//if no unique element is found;* }  
  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Enter array size");  
 int size = sc.nextInt();  
  
 int arr[] = new int[size];  
 System.*out*.println("Enter elements of Array:");  
 for (int i = 0; i < size; i++) {  
 arr[i] = sc.nextInt();  
 }  
  
  
 int Unique\_number = *findFirstUnique*(arr);  
 if (Unique\_number != -1) {  
 System.*out*.println("First Unique number :" + Unique\_number);  
 } else {  
 System.*out*.println("unique number not found");  
 }  
  
}  
 }

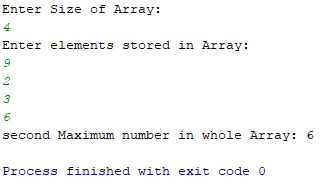
**OUTPUT:**

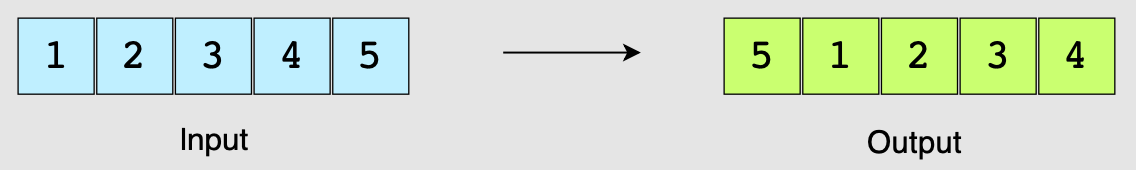
8: In this problem, you have to implement the **int findSecondMaximum(int[] arr)** method, which will traverse the whole array and return the second largest element present in the array.

**Assumption:** Array should contain at least two unique elements.

package FirstLabTasks;  
  
import java.util.Scanner;  
public class SecondMaximumNumber {  
  
 public static int findSecondMaximum(int[] arr) {  
 int max = arr[0];  
 int secondMax = arr[1];  
  
 if (secondMax > max) {  
 int temp = max;  
 max = secondMax;  
 secondMax = temp;  
 }  
 for (int i = 2; i < arr.length; i++) {  
 if (arr[i] > max) {  
 secondMax = max;  
 max = arr[i];  
 } else if (arr[i] > secondMax && arr[i] != max) {  
 secondMax = arr[i];  
 }  
 }  
  
 return secondMax;  
 }  
  
 public static void main(String[] args) {  
  
  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Enter Size of Array:");  
 int Arr\_Size = sc.nextInt();  
  
 if (Arr\_Size< 2) {  
 System.*out*.println("Array should contain at least two unique elements.");  
 sc.close();  
 return;  
 }  
  
 int[] arr = new int[Arr\_Size];  
 System.*out*.println("Enter elements stored in Array:");  
  
 for (int i = 0; i < Arr\_Size; i++) {  
 arr[i] = sc.nextInt();  
 }  
  
 int secondMax = *findSecondMaximum*(arr);  
 System.*out*.println("second Maximum number in whole Array: " + secondMax);  
  
  
 }  
 }

**OUTPUT:**



9:I this problem, you have to implement the **void rotateArray(int[] arr)** method, which takes an arr and rotate it right by 1. This means that the right-most elements will appear at the left-most position in the arrayn.

package FirstLabTasks;  
  
import java.util.Scanner;  
  
public class RightRotateArray {

public static void rotateArray(int[] arr) {

if (arr == null || arr.length <= 1) {  
  
 return;  
 }

int lastElement = arr[arr.length - 1]; *// Store the last element* for (int i = arr.length - 1; i > 0; i--) {  
 arr[i] = arr[i - 1]; *// Shift elements to the right* }

arr[0] = lastElement; *// Place the last element at the beginning* }

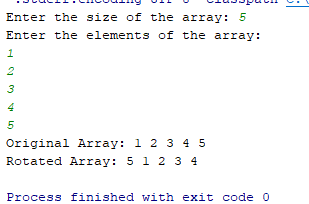
public static void main(String[] args) {

Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the size of the array: ");  
 int size = scanner.nextInt();

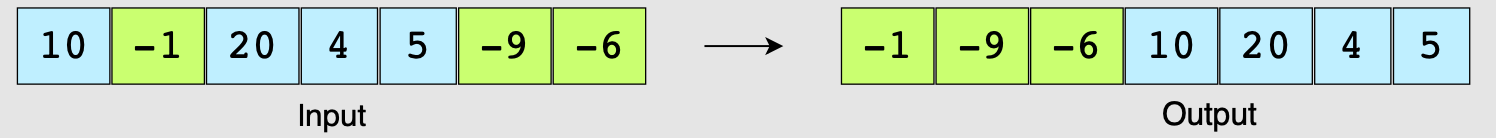
int[] arr = new int[size];  
 System.*out*.println("Enter the elements of the array:");  
 for (int i = 0; i < size; i++) {  
 arr[i] = scanner.nextInt();  
 }

System.*out*.print("Original Array: ");  
 for (int i = 0; i < size; i++) {  
 System.*out*.print(arr[i] + " ");  
 }  
 System.*out*.println();  
  
 *rotateArray*(arr);  
  
 System.*out*.print("Rotated Array: ");  
 for (int i = 0; i < size; i++) {  
 System.*out*.print(arr[i] + " ");  
 }  
 System.*out*.println();  
  
  
 }  
 }

**OUTPUT**



**10:** In this problem, you have to implement the **void reArrange(int[] arr)** method, which will sort the elements, such that all the negative elements appear at the left and positive elements appear at the right.



package FirstLabTasks;  
  
import java.util.Scanner;  
  
public class ArrayReArrange {  
  
  
 public static void reArrange(int[] arr) {  
 int left = 0;  
 int right = arr.length - 1;  
  
 while (left <= right) {  
  
 while (left < arr.length && arr[left] < 0) {  
 left++;  
 }  
  
 while (right >= 0 && arr[right] >= 0) {  
 right--;  
 }  
  
  
 if (left < right) {  
 int temp = arr[left];  
 arr[left] = arr[right];  
 arr[right] = temp;  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the size of the array: ");  
 int size = scanner.nextInt();

int[] arr = new int[size];

System.*out*.println("Enter the elements of the array:");  
 for (int i = 0; i < size; i++) {  
 arr[i] = scanner.nextInt();  
 }  
  
 *reArrange*(arr);  
 System.*out*.println("Rearranged array:");

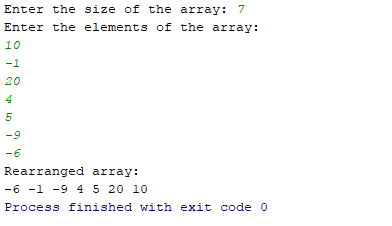
for (int i = 0; i < arr.length; i++) {

System.*out*.print(arr[i] + " ");

}

}  
  
  
}

**OUTPUT**

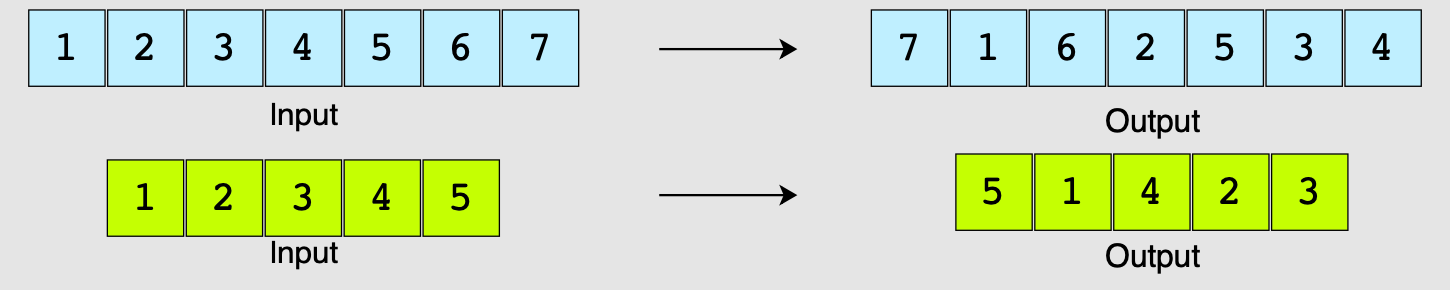


**11:**In this problem, you have to implement the **void maxMin(int[] arr)** method. This will

re-arrange the elements of a sorted array in such a way that the first position will have the

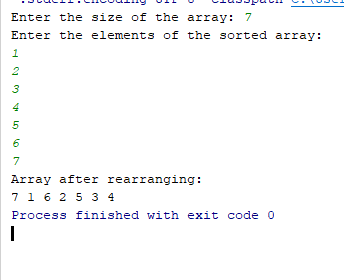
largest number, the second will have the smallest, the third will have the second-largest, and so on.

**Note:** The given array is sorted in ascending order.

**Note:** The range of integers in the array can be from 0 to 10000.

package FirstLabTasks;  
  
import java.util.Scanner;  
public class MaximumMinimumArrange {  
  
 public static void maxMin(int[] arr) {  
 int n = arr.length;  
 int[] result = new int[n];  
 int left = 0;  
 int right = n - 1;  
  
 for (int i = 0; i < n; i++) {  
 if (i % 2 == 0)  
 result[i] = arr[right--];  
 else  
 result[i] = arr[left++];  
 }  
  
 for (int i = 0; i < n; i++) {  
 arr[i] = result[i];  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the size of the array: ");  
 int n = scanner.nextInt();  
  
 int[] arr = new int[n];  
  
 System.*out*.println("Enter the elements of the sorted array:");  
 for (int i = 0; i < n; i++) {  
 arr[i] = scanner.nextInt();  
 }  
  
 *maxMin*(arr);  
  
 System.*out*.println("Array after rearranging:");  
 for (int i = 0; i < arr.length; i++) {  
 System.*out*.print(arr[i] + " ");  
 }  
 }  
 }

**OUTPUT**



**12:** Create an function int **findMaxSumSubArray(int[] arr)** that will take an integer array, will return the maximum subarray sum. The array may contain both positive and negative integers and is unsorted.

**Sample Input:** arr = {1, 7, -2, -5, 10, -1}

# Output: 11

# Explanation of Question:

# Here is exaplanation of what is mean by subarraysum and MaxSubArraySum

# 

# \\

# CODE :

package FirstLabTasks;  
  
import java.util.Scanner;  
  
public class MaximumSubArray {  
  
 public static int findMaxSumSubArray(int[] arr){  
  
  
 int maxSum=arr[0];  
 int Sum=arr[0];  
  
 for(int i=1; i<arr.length; i++){  
  
 if(Sum>=0){  
 Sum+=arr[i];  
 }  
  
 else {  
  
 Sum=arr[i];  
 }  
  
 if(Sum>maxSum){  
 maxSum=Sum;  
  
 }  
 }  
  
 return maxSum;  
  
 }  
  
  
  
 public static void main(String[] args) {  
  
 Scanner sc=new Scanner(System.*in*);  
 System.*out*.println("What is the size of Array?");  
 int size=sc.nextInt();  
  
 int arr[]=new int[size];  
 System.*out*.println("Enter elements :");  
 for(int i=0; i<size; i++){  
  
 arr[i]=sc.nextInt();  
  
 }

int MaxSumValue=*findMaxSumSubArray*(arr);  
 System.*out*.print("Max Sum is :"+MaxSumValue);  
  
  
 }  
}

**OUTPUT:**

