**1)READ ARRAY ELEMENTS FROM USER**

**package** com.ArraysPrograms;

**import** java.util.Scanner;

**public** **class** ReadArray {

// READ ARRAY FROM USER

**static** **int**[]readArray(){

Scanner sc=**new** Scanner(System.***in***);

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

**int** n=sc.nextInt();

**int** arr[]=**new** **int**[n];

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

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

arr[i]=sc.nextInt();

}

**return** arr;

}

// DISPLAY ARRAY ELEMENTS

**static** **void** displayArray(**int** arr[]) {

System.***out***.println("Array elements are");

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

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

**if**(i<arr.length-1)

System.***out***.print(",");

}

System.***out***.println();

}

//BIGGEST ELEMENT FROM ARRAY

**static** **int** isBiggest(**int** arr[]) {

**int** big=arr[0];

**for**(**int** i=1;i<arr.length;i++) {

**if**(arr[i]>big)

big=arr[i];

}

**return** big;

}

// SMALLEST ELEMENT FROM ARRAY

**static** **int** isSmallest(**int** arr[]) {

**int** small=arr[0];

**for**(**int** i=1;i<arr.length;i++) {

**if**(arr[i]<small)

small=arr[i];

}

**return** small;

}

//SUM OF ARRAY ELEMENTS

**static** **int** isSum(**int** arr[]) {

**int** sum=0;

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

sum=sum+arr[i];

}

**return** sum;

}

//EVEN AND ODD NUMBER FROM ARRAY

**static** **void** isEvenOdd(**int** arr[]) {

**int** even=0,odd=0;

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

**if**(arr[i]%2==0)

even++;

**else**

odd++;

}

System.***out***.println("Even number count "+even);

System.***out***.println("Odd number count "+odd);

}

**public** **static** **void** main(String[] args) {

ReadArray ra=**new** ReadArray();

**int**[] rd=ra.*readArray*();

ra.*displayArray*(rd);

**int** b=*isBiggest*(rd);

System.***out***.println("Biggest element of array is "+b);

**int** s=*isSmallest*(rd);

System.***out***.println("Smallest element of array is "+s);

**int** s1=*isSum*(rd);

System.***out***.println("Sum of array elements is "+s1);

*isEvenOdd*(rd);

}

}

**2)FIND ARRAYS ELEMENTS FREQUENCY**

**package** com.ArraysPrograms;

//FIND ARRAYS ELEMENTS FREQUENCY

**public** **class** ArrayElementFrequency

{

**static** **void** countFreq(**int** arr[])

{

**boolean** rs[]=**new** **boolean**[arr.length];

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

{

**if**(rs[i]==**false**)

{

**int** count=1;

**for**(**int** j=i+1;j<arr.length;j++)

{

**if**(arr[i]==arr[j])

{

count++;

rs[j]=**true**;

}

}

System.***out***.println(arr[i]+"-->"+count);

}

}

}

**public** **static** **void** main(String[] args) {

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

*countFreq*(arr);

}

}

**3)FIND COMMON ELEMENTS BETWEEN TWO ARRAYS**

**package** com.ArraysPrograms;

//FIND COMMON ELEMENTS BETWEEN TWO ARRAYS

**public** **class** ArraysCommonElements

{

**public** **static** **void** main(String[] args)

{

**int** arr1[]= {10,20,30,40};

**int** arr2[]= {30,40,50,10};

**int** ce[]=*interSection*(arr1,arr2);

System.***out***.println("Common elements are ");

**for**(**int** i=0;i<ce.length;i++)

{

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

}

System.***out***.println();

}

**static** **int**[] interSection(**int**[]a, **int**[] b) {

**int** c[];

**if**(a.length<b.length)

c=**new** **int**[a.length];

**else**

c=**new** **int**[b.length];

**int** k=0;

**for**(**int** i=0;i<a.length;i++)

{

**for**(**int** j=0;j<b.length;j++)

{

**if**(a[i]==b[j])

{

c[k]=a[i];

k++;

**break**;

}

}

}

**int** res[]=**new** **int**[k];

**for**(**int** i=0;i<res.length;i++)

{

res[i]=c[i];

}

**return** res;

}

}

**4)SORT THE ARRAY ELEMENTS**

**package** com.ArraysPrograms;

//SORT THE ARRAY ELEMENTS

**public** **class** ArraySort

{

**static** **int** [] sortArray(**int** arr[])

{

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

{

**for**(**int** j=i+1;j<arr.length;j++)

{

**int** temp=0;

**if**(arr[i]>arr[j])

{

temp = arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

**return** arr;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {20,50,10,40,30};

*sortArray*(arr);

**int**[] sa=*sortArray*(arr);

System.***out***.println("After sort arrays elements are");

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

System.***out***.println(sa[i]+" ");

}

}

}

**5)COPY ARRAY ELEMENTS**

**package** com.ArraysPrograms;

// COPY ARRAY ELEMENTS

**public** **class** CopyArray {

// static int[] copyArray(int arr1[]) {

// int arr2[]=new int[arr1.length];

// for(int i=0;i<arr1.length;i++) {

// arr2[i]=arr1[i];

// }

// return arr2;

// }

// **REVERSE THE ARRAY**

**static** **void** reverseArray(**int** arr[]) {

**int** l=0,h=arr.length-1;

**while**(l<h) {

**int** temp=arr[l];

arr[l]=arr[h];

arr[h]=temp;

l++;

h--;

}

System.***out***.println("Reverse array is");

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

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

}

System.***out***.println();

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5};

// System.out.println("Array 2 elements are");

// int c[]=copyArray(arr1);

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

// System.out.println(c[i]+" ");

*reverseArray*(arr);

}

}

**6)INSERT AN ELEMENT IN A SPECIFIED POSITION IN AN ARRAY**

**package** com.ArraysPrograms;

// INSERT AN ELEMENT IN A SPECIFIED POSITION IN AN ARRAY

**public** **class** InsertElement

{

**static** **int**[]insertElement(**int** []arr,**int** ele,**int** in)

{

**if**(in<0||in>arr.length)

{

System.***out***.println("Index out of range");

**return** arr;

}

**int**[]arr2=**new** **int**[arr.length+1];

arr2[in]=ele;

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

{

**if**(i<in)

arr2[i]=arr[i];

**else**

arr2[i+1]=arr[i];

}

**return** arr2;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {10,20,30,50,60};

**int** []ia=*insertElement*(arr,40,5);

//System.out.println("After inserting elements array becomes");

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

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

}

System.***out***.println();

}

}

**7)DELETE AN ELEMENT IN A SPECIFIED POSITION IN AN ARRAY**

**package** com.ArraysPrograms;

//DELETE AN ELEMENT IN A SPECIFIED POSITION IN AN ARRAY

**public** **class** DeleteElement {

**static** **int**[]deleteElement(**int** []x,**int** in)

{

**if**(in<0||in>=x.length)

{

System.***out***.println("Index is not range");

**return** x;

}

**int** y[]=**new** **int**[x.length-1];

**for**(**int** i=0;i<y.length;i++)

{

**if**(i<in)

y[i]=x[i];

**else**

y[i]=x[i+1];

}

**return** y;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {10,20,30,40,50};

**int** []da=*deleteElement*(arr,5);

//System.out.println("After deleting elements array becomes");

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

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

}

System.***out***.println();

}

}

**8)COUNT EVEN AND ODD NUMBER IN ARRAY**

**package** com.ArraysPrograms;

**public** **class** EvenOddCount {

//COUNT EVEN AND ODD NUMBER IN ARRAY

**static** **void** isEvenOdd(**int** arr[]) {

**int** even=0,odd=0;

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

**if**(arr[i]%2==0)

even++;

**else**

odd++;

}

System.***out***.println("Even count is "+even);

System.***out***.println("Odd count is "+odd);

}

**//PRINT EVEN AND ODD ELEMENTS SUM IN ARRAY**

**static** **int**[]printEvenOdd(**int** arr[]){

**int** esum=0,osum=0;

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

**if**(arr[i]%2==0)

esum=esum+arr[i];

**else**

osum=osum+arr[i];

}

**int** sum[]= {esum,osum};

**return** sum;

}

**//COUNT PRIME NUMBER IN GIVEN ARRAY**

**static** **int** countPrime(**int** arr[]) {

**int** pcount=0;

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

**if**(*isPrime*(arr[i]))

pcount++;

}

**return** pcount;

}

**static** **boolean** isPrime(**int** n) {

**if**(n<=1)

**return** **false**;

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

**if**(n%i==0)

**return** **false**;

}

**return** **true**;

}

**// COUNT SPECIAL TWO DIGIT NUMBER IN AN ARRAY**

**static** **int** countSpecialDigit(**int** arr[]) {

**int** count=0;

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

**if**(*isSpecialDigit*(arr[i]))

count++;

}

**return** count;

}

**static** **boolean** isSpecialDigit(**int** n) {

**int** d1=n/10;

**int** d2=n%10;

**return** n==d1+d2+d1\*d2;

}

**// COUNT POSITIVE AND NEGATIVE NUMBER IN ARRAY**

**static** **void** isPosNeg(**int** arr[]) {

**int** posc=0,negc=0;

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

**if**(arr[i]>0)

posc++;

**else**

negc++;

}

System.***out***.println("Total positive count "+posc);

System.***out***.println("Total negative count "+negc);

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,-3,4};

*isEvenOdd*(arr);

**int** []s=*printEvenOdd*(arr);

System.***out***.println("Even sum is "+s[0]);

System.***out***.println("Odd sum is "+s[1]);

**int** p=*countPrime*(arr);

System.***out***.println("Prime count is "+p);

**int** c=*countSpecialDigit*(arr);

System.***out***.println("Total special digit count is "+c);

*isPosNeg*(arr);

}

}

**9)NTH LARGEST ELEMENT IN AN ARRAY**

**package** com.ArraysPrograms;

// NTH LARGEST ELEMENT IN AN ARRAY

**public** **class** FindNthBigElement {

**static** **void** isLargest(**int** arr[],**int** n) {

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

**for**(**int** j=i+1;j<arr.length;j++) {

**if**(arr[i]<arr[j]) {

**int** temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

**if**(i==n-1)

System.***out***.print(n+"th largest element is "+arr[i]);

**break**;

}

}

**public** **static** **void** main(String[] args) {

**int** arr1[] ={1,2,3,4,5};

*isLargest*(arr1,4);

}

}

**10)FIND NTH SMALLEST ELEMENT IN AN ARRAY**

**package** com.ArraysPrograms;

// FIND NTH SMALLEST ELEMENT IN AN ARRAY

**public** **class** FindNthSmallElement {

**static** **void** isSmallest(**int** arr[],**int** n) {

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

**for**(**int** j=i+1;j<arr.length;j++) {

**if**(arr[i]>arr[j]) {

**int** temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

**if**(i==n-1) {

System.***out***.println(n+"th smallest element is "+arr[i]);

**break**;

}

}

}

**public** **static** **void** main(String[] args) {

**int** arr[] ={5,4,3,2,1};

*isSmallest*(arr,1);

}

}

**11)MERGE TWO SORTED ARRAYS IN A SORTED MANNER**

**package** com.ArraysPrograms;

**import** java.util.Arrays;

// MERGE TWO SORTED ARRAYS IN A SORTED MANNER

**public** **class** MergeSortArray {

**static** **int**[] isMerge(**int** x[],**int** y[]) {

**int** z[]=**new** **int**[x.length+y.length];

**int** i=0,j=0,k=0;

**while**(i<x.length&&j<y.length) {

**if**(x[i]<y[j])

z[k++]=x[i++];

**else**

z[k++]=y[j++];

**while**(i<x.length) {

z[k++]=x[i++];

}

}

**while**(j<y.length) {

z[k++]=y[j++];

}

**return** z;

}

**public** **static** **void** main(String[] args) {

**int** arr1[]= {1,2,3,4};

**int** arr2[]= {5,6,7,8};

**int**[]ma=*isMerge*(arr1,arr2);

System.***out***.println("Merge array in a sorted manner");

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

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

}

System.***out***.println();

}

}

**12)MERGE TWO ARRAYS ELEMENTS**

**package** com.ArraysPrograms;

// MERGE TWO ARRAYS ELEMENTS

**public** **class** MergeTwoArrays {

**static** **int** []mergeArray(**int** x[],**int** y[]){

**int** z[]=**new** **int**[x.length+y.length];

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

z[i]=x[i];

}

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

z[x.length+i]=y[i];

}

**return** z;

}

**public** **static** **void** main(String[] args) {

**int** [] arr1= {10,30,20,15,40};

**int**[] arr2= {50,25,60};

**int**[]ma=*mergeArray*(arr1,arr2);

System.***out***.println("After merge the array");

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

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

}

System.***out***.println();

}

}

**13)SEARCH AN ELEMENT FROM AN ARRAY**

**package** com.ArraysPrograms;

//SEARCH AN ELEMENT FROM AN ARRAY

**public** **class** SearchElement {

**static** **int** isSearch(**int** arr[],**int** ele) {

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

**if**(arr[i]==ele)

**return** i;

}

**return** -1;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {10,20,30,40};

**int** se=*isSearch*(arr,30);

System.***out***.println("elements in at index "+se);

}

}

**14)MERGE TWO ARRAYS ELEMENTS IN A SINGLE ARRAY IN ZIGZAG ORDER**

**package** com.ArraysPrograms;

//MERGE TWO ARRAYS ELEMENTS IN A SINGLE ARRAY IN ZIGZAG ORDER

//ARR1[]={10,20,30,40}; ARR2[]={50,60,70};

//ARR3[]={10,50,20,60,30,70,40};

**public** **class** ZagzigOrder {

**static** **int** []isZagzig(**int** x[],**int** y[]){

**int** z[]=**new** **int**[x.length+y.length];

**int** i=0;

**int** k=0;

**while**(i<x.length&&i<y.length) {

z[k++]=y[i];

//k++;

z[k++]=x[i++];

//k++;

//i++;

}

**while**(i<y.length) {

z[k++]=y[i++];

// k++;

// i++;

}

**while**(i<x.length) {

z[k++]=x[i++];

// k++;

// i++;

}

**return** z;

}

**public** **static** **void** main(String[] args) {

**int** arr1[]= {10,20,30,40,50};

**int** arr2[]= {60,70,80,90,100};

System.***out***.println("Array elements after zagzig orders");

**int** []za=*isZagzig*(arr1,arr2);

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

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

}

System.***out***.println();

}

}

**15)ARRAYS ELEMENTS SUM**

**package** com.ArrayPrograms;

//ARRAYS ELEMENTS SUM

**public** **class** ArraySum {

**static** **int** *arr*[]= {10,20,30,40};

**static** **int** isSum() {

**int** sum=0;

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

sum=sum+*arr*[i];

}

**return** sum;

}

**public** **static** **void** main(String[] args) {

System.***out***.println("Arrays elements sum is "+*isSum*());

}

}

**16)COUNT PALINDROME NUMBER IN AN GIVEN ARRAY**

**package** com.ArrayPrograms;

// COUNT PALINDROME NUMBER IN AN GIVEN ARRAY

**public** **class** CountPalindromeNumber {

**static** **int** countPalindrome(**int** arr[]) {

**int** count=0;

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

**if**(*isPalindrome*(arr[i]))

count++;

}

**return** count;

}

**static** **boolean** isPalindrome(**int** n) {

**int** rev=0;

**int** temp=n;

**while**(n!=0) {

**int** rem=n%10;

rev=rev\*10+rem;

n=n/10;

}

**if**(rev==temp)

**return** **true**;

**else**

**return** **false**;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {10,11,12,22,17};

**int** ct=*countPalindrome*(arr);

System.***out***.println("Total palindrome number is "+ct);

}

}

**17)EVEN AND ODD NUMBER SUM IN AN ARRAY**

**package** com.ArrayPrograms;

//EVEN AND ODD NUMBER SUM IN AN ARRAY

**public** **class** EvenOddSum {

**static** **int**[] isEvenOdd(**int** arr[]) {

**int** esum=0,osum=0;

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

**if**(arr[i]%2==0)

esum=esum+arr[i];

**else**

osum=osum+arr[i];

}

**int**[]sum= {esum,osum};

**return** sum;

}

**public** **static** **void** main(String[] args) {

**int**[]arr= {10,15,20,25};

**int**[] total\_sum=*isEvenOdd*(arr);

System.***out***.println("Even number sum is "+total\_sum[0]);

System.***out***.println("Odd number sum is "+total\_sum[1]);

}

}

**18)COUNT POSITIVE & NEGATIVE NUMBER IN AN ARRAY**

**package** com.ArrayPrograms;

// COUNT POSITIVE & NEGATIVE NUMBER IN AN ARRAY

**public** **class** PosNegNumber {

**static** **int**[]isPosNeg(**int** arr[]) {

**int** pc=0,nc=0;

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

**if**(arr[i]>0)

pc++;

**else**

nc++;

}

**int** []ct= {pc,nc};

**return** ct;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {10,-20,30,-40};

**int**[]count=*isPosNeg*(arr);

System.***out***.println("Positive number count is "+count[0]);

System.***out***.println("Negative number count is "+count[1]);

}

}

**19)COUNT TOTAL PRIME NUMBER IN A GIVEN ARRAY**

**package** com.ArrayPrograms;

// COUNT TOTAL PRIME NUMBER IN A GIVEN ARRAY

**public** **class** PrimeNum {

**static** **boolean** isPrime(**int** n) {

**if**(n==1)

**return** **false**;

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

**if**(n%i==0)

**return** **false**;

}

**return** **true**;

}

**static** **int** countPrime(**int** arr[]) {

**int** count=0;

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

**if**(*isPrime*(arr[i]))

count++;

}

**return** count;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5};

**int** count=*countPrime*(arr);

System.***out***.println("Total prime numbers are "+count);

}

}

**20)REPLACING ALL ARRAYS ELEMENTS BY ITS DIGIT SUM**

**package** com.ArrayPrograms;

//REPLACING ALL ARRAYS ELEMENTS BY ITS DIGIT SUM

**public** **class** ReplaceBySum {

**static** **int**[] isSum(**int** arr[]) {

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

arr[i]=*isDigitSum*(arr[i]);

}

**return** arr;

}

**static** **int** isDigitSum(**int** n) {

**int** sum=0;

**while**(n!=0) {

sum=sum+n%10;

n=n/10;

}

**return** sum;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {101,120,930,409};

**int** ds[]=*isSum*(arr);

System.***out***.println("Replacing array is");

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

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

}

System.***out***.println();

}

}

**21)COUNT TOTAL SPECIAL NUMBER PRESENT IN AN ARRAY**

**package** com.ArrayPrograms;

//COUNT TOTAL SPECIAL NUMBER PRESENT IN AN ARRAY

**public** **class** SpecialNumArray {

**static** **int** isSpecial(**int** arr[]) {

**int** count=0;

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

**int** d1=arr[i]%10;

**int** d2=arr[i]/10;

**int** sum=(d1\*d2)+(d1+d2);

**if**(sum==arr[i])

count++;

}

**return** count;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {29,20,59,60};

System.***out***.println("Total Special number count is "+*isSpecial*(arr));

}

}

**22)ARRAY ELEMENTS RIGHT ROTATION BY N POSITION**

**package** com.ArraysPrograms;

//ARRAY ELEMENTS RIGHT ROTATION BY N POSITION

**public** **class** ArrayRightRotation {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5,6,7};

**int** n=2;

**int** rr[]=*rightRotation*(arr,n);

System.***out***.println("After right rotation");

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

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

}

System.***out***.println();

}

**static** **int**[] rightRotation(**int**[] arr,**int** k) {

**while**(k>0) {

**int** last=arr[arr.length-1];

**for**(**int** i=arr.length-1;i>0;i--) {

arr[i]=arr[i-1];

}

arr[0]=last;

k--;

}

**return** arr;

}

}

**23)ARRAY ELEMENTS LEFT ROTATION BY N POSITION**

**package** com.ArraysPrograms;

// ARRAY ELEMENTS LEFT ROTATION BY N POSITION

**public** **class** ArrayLeftRotation {

**static** **int**[] leftRotation(**int**[] arr, **int** k) {

**while**(k>0) {

**int** first=arr[0];

**for**(**int** i=1;i<arr.length;i++) {

arr[i-1]=arr[i];

}

arr[arr.length-1]=first;

k--;

}

**return** arr;

}

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5,6,7};

**int** n=2;

**int** lr[]=*leftRotation*(arr,n);

System.***out***.println("After left rotation");

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

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

}

System.***out***.println();

}

}

**24)FIND SECOND LARGEST ELEMENT IN AN ARRAY**

**package** com.ArrayPrograms;

//FIND SECOND LARGEST ELEMENT IN AN ARRAY

**public** **class** SecondLargestElementArray {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5};

**int** se=*secondLargest*(arr);

System.***out***.println("Second Largest Element is "+se);

}

**private** **static** **int** secondLargest(**int**[] arr) {

**int** max1=0;

**int** max2=0;

**if**(arr[0]>max1) {

max1=arr[0];

max2=arr[1];

}

**else** {

max1=arr[1];

max2=arr[0];

}

**for**(**int** i=2;i<arr.length;i++) {

**if**(arr[i]>max1) {

max2=max1;

max1=arr[i];

}

**else** **if**(arr[i]>max2) {

max2=arr[i];

}

}

**return** max2;

}

}

**25)FIND ALL THE SUB-ARRAY OF AN ARRAY**

**package** com.ArrayPrograms;

//FIND ALL THE SUB-ARRAY OF AN ARRAY

//A sub-array is a contiguous part of array. An array that is inside another array.

**public** **class** FindSubArray {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5};

System.***out***.println("All non empty subarrays are");

*findSubArray*(arr);

}

**private** **static** **void** findSubArray(**int**[] arr) {

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

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

**for**(**int** k=i;k<=j;k++) {

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

}

System.***out***.println();

}

}

}

}

**26)FIND THE LEADER OF AN ARRAY**

**package** com.ArrayPrograms;

//FIND THE LEADER OF AN ARRAY

//An element is a leader if it is greater than all the elements to its right side.

//And the rightmost element is always a leader.

**public** **class** ArrayLeaders {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,9,10,4,5};

*leaderArray*(arr);

// arrayLeaders(arr);

}

// APPROACH-2

**private** **static** **void** leaderArray(**int**[] arr) {

**int** curr\_leader=arr[arr.length-1];

System.***out***.println(curr\_leader);

**for**(**int** i=arr.length-2;i>=0;i--) {

**if**(arr[i]>curr\_leader)

curr\_leader=arr[i];

}

System.***out***.println(curr\_leader);

}

// APPROACH-1

/\* private static void arrayLeaders(int[] arr) {

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

boolean isleader=true;

for(int j=i+1;j<arr.length;j++) {

if(arr[j]>arr[i]) {

isleader=false;

break;

}

}

if(isleader==true)

System.out.println(arr[i]);

}

}\*/

}

**27)MOVE ALL THE ZEROS TO THE END OF THE ARRAY**

**package** com.ArrayPrograms;

//MOVE ALL THE ZEROS TO THE END OF THE ARRAY

//ARR[]={1,0,2,0,3,0,4};

//OUTPUT-->ARR1[]={1,2,3,4,0,0,0};

**public** **class** MoveAllZerosToEnd {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,0,2,0,3,0,4};

**int** mz[]=*moveZeroEnd*(arr);

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

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

}

System.***out***.println();

}

**private** **static** **int**[] moveZeroEnd(**int**[] arr) {

**if**(arr.length==0||arr.length==1) {

**return** arr;

}

**int** z=0;

**int** nz=0;

**while**(nz<arr.length) {

**if**(arr[nz]!=0) {

**int** temp=arr[nz];

arr[nz]=arr[z];

arr[z]=temp;

nz++;

z++;

}

**else**

nz++;

}

**return** arr;

}

}

**28)MOVE ALL THE ZEROS TO THE FIRST OF THE ARRAY**

**package** com.ArrayPrograms;

//MOVE ALL THE ZEROS TO THE FIRST OF THE ARRAY

**public** **class** MoveAllZerosToStart {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,0,2,0,3,0};

**int** mz[]=*moveZeroFirst*(arr);

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

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

}

System.***out***.println();

}

**private** **static** **int**[] moveZeroFirst(**int**[] arr) {

**if**(arr.length==0||arr.length==1) {

**return** arr;

}

**int** z=arr.length-1;

**int** nz=arr.length-1;

**while**(nz>=0) {

**if**(arr[nz]!=0) {

**int** temp=arr[nz];

arr[nz]=arr[z];

arr[z]=temp;

nz--;

z--;

}

**else**

nz--;

}

**return** arr;

}

}

**29)Maximum AND value of a pair in an array**

**package** com.ArrayPrograms;

//Maximum AND value of a pair in an array

**public** **class** MaximumAndValuePair {

**public** **static** **void** main(String[] args) {

**int** arr[]= {16,9,6,13};

**int** mv=*maxAndValue*(arr);

System.***out***.println("Maximum and value "+mv);

}

**private** **static** **int** maxAndValue(**int**[] arr) {

**int** res=0;

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

**for**(**int** j=i+1;j<arr.length;j++) {

// if((arr[i]&arr[j])>res)

// res=arr[i]&arr[j];

res=res>(arr[i]&arr[j])?res:(arr[i]&arr[j]);

}

}

**return** res;

}

}

**30)Find a pair with maximum product in array of Integers**

**package** com.ArrayPrograms;

//Find a pair with maximum product in array of Integers

**public** **class** MaximumPoductArray {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5};

*maxProduct*(arr);

}

**private** **static** **void** maxProduct(**int**[] arr) {

**if**(arr.length<2) {

System.***out***.println("Product pair not exists");

**return**;

}

**int** a=arr[0];

**int** b=arr[1];

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

{

**for**(**int** j=i+1;j<arr.length;j++)

{

**if**(arr[i]\*arr[j]>a\*b)

{

a=arr[i];

b=arr[j];

}

}

}

System.***out***.println("Max product pair is {" +

a + ", " + b + "}");

System.***out***.println("Maximum product is "+(a\*b));

}

}

**31)LINEAR SEARCH ON GIVEN ARRAYS ELEMENTS**

**package** com.ArrayPrograms;

//LINEAR SEARCH

we use linear search in following things:

• for search item in the smaller array.

• For fast searching

The time complexity of the above algorithm is O(n).

**public** **class** LinearSearch {

**public** **static** **void** main(String[] args) {

**int** arr[]= {10,20,30,40};

**int** key=60;

System.***out***.print("Key Element is found at index ");

System.***out***.println(*linearSearch*(arr,key));

}

**private** **static** **int** linearSearch(**int**[] arr,**int** key) {

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

**if**(arr[i]==key)

**return** i;

}

**return** -1;

}

}

**32)BINARY SEARCH ON GIVEN ARRAYS ELEMENTS**

**package** com.ArraysPrograms;

**import** java.util.Arrays;

/\*Binary Search is defined as a searching algorithm used in a sorted array

by repeatedly dividing the search interval in half.

The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(log N).

\*/

**public** **class** BinarySearch {

**public** **static** **void** main(String[] args) {

**int** arr[]= {10,20,30,40,50};

**int** key=10;

System.***out***.println(*binarySearch*(arr,key));

}

**private** **static** **int** binarySearch(**int**[] arr, **int** key) {

**int** l=0;

**int** h=arr.length-1;

**int** mid=0;

**while**(l<=h) {

mid=l+h/2;

**if**(key==arr[mid])

**return** mid;

**else** **if**(key<arr[mid])

h=mid-1;

**else**

l=mid+l;

}

**return** -1;

}

}

**33)FIND CEIL AND FLOOR OF A NUMBER IN A SORTED ARRAY**

**package** com.ArraysPrograms;

/\*Given an unsorted/sorted array arr[] and an element x, find floor and ceiling of x in arr[0..n-1].

Floor of x is the largest element which is smaller than or equal to x.

Floor of x doesn’t exist if x is smaller than smallest element of arr[].

Ceil of x is the smallest element which is greater than or equal to x.

Ceil of x doesn’t exist if x is greater than greatest element of arr[].

\*/

**public** **class** FindCeilAndFloorNumber {

**public** **static** **void** main(String[] args) {

**int** arr[]= {10,20,30,40,50};

**int** key=40;

System.***out***.println(*findCeil*(arr,key));

System.***out***.println(*findFloor*(arr,key));

}

**private** **static** **int** findFloor(**int**[] arr, **int** key) {

**int** low=0;

**int** high=arr.length-1;

**int** mid=0;

**while**(low<=high)

{

mid=(low+high)/2;

**if**(key==arr[mid])

{

**return** arr[mid];

}

**else** **if**(key<arr[mid])

{

high=mid-1;

}

**else**

{

low=mid+1;

}

}

**if**(high>=0)

{

**return** arr[high];

}

**else**

{

**return** -1;

}

}

**private** **static** **int** findCeil(**int**[] arr, **int** key) {

**int** low=0;

**int** high=arr.length-1;

**int** mid=0;

**while**(low<=high)

{

mid=(low+high)/2;

**if**(key==arr[mid])

{

**return** arr[mid];

}

**else** **if**(key<arr[mid])

{

high=mid-1;

}

**else**

{

low=mid+1;

}

}

**if**(low<arr.length)

{

**return** arr[low];

}

**else** {

**return** -1;

}

}

}

**34)FIND FAST AND LAST POSITION OF AN ELEMENT IN A SORTED ARRAY**

**package** com.ArraysPrograms;

//FIND FAST AND LAST POSITION OF AN ELEMENT IN A SORTED ARRAY

**public** **class** FastAndLastPosition {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5,5};

**int** key=5;

**int** fl[]=*searchRange*(arr,key);

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

System.***out***.println(fl[i]+" ");

}

}

**private** **static** **int**[] searchRange(**int**[] arr, **int** key) {

**int** res[]= {-1,-1};

**int** low=0;

**int** high=arr.length-1;

**int** mid=0;

**while**(low<=high)

{

mid=(low+high)/2;

**if**(key==arr[mid])

{

res[0]=mid;

high=mid-1;

}

**else** **if**(key>arr[mid])

{

low=mid+1;

}

**else**

{

high=mid-1;

}

}

low=0;

high=arr.length-1;

mid=0;

**while**(low<=high)

{

mid=(low+high)/2;

**if**(key==arr[mid])

{

res[1]=mid;

low=mid+1;

}

**else** **if**(key>arr[mid])

{

low=mid+1;

}

**else**

{

high=mid-1;

}

}

**return** res;

}

}

**35)Majority Element Using Moore’s Voting Algorithm**

**package** com.ArraysPrograms;

//Majority Element Using Moore’s Voting Algorithm:

/\*A majority element in an array A[] of size n is an element that appears more than n/2 times

(and hence there is at most one such element).\*/

**public** **class** FindMajorityElementInArray {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5,5,5};

**int** fm=*findMajorityElement*(arr);

System.***out***.println("Majority Element is "+fm);

}

**private** **static** **int** findMajorityElement(**int**[] arr)

{

**int** maj=arr[0];

**int** count=1;

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

{

**if**(arr[i]==maj)

{

count++;

}

**else**

{

count--;

}

**if**(count==0)

{

maj=arr[i];

count=1;

}

}

**return** maj;

}

}

**36)Find the majority element in the array.**

**package** com.ArraysPrograms;

//Find the majority element in the array.

**public** **class** FindMajority {

**public** **static** **void** main(String[] args) {

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

**int** n = arr.length;

*findMajority*(arr, n);

}

**private** **static** **void** findMajority(**int**[] arr, **int** n) {

**int** maxCount = 0;

**int** index = -1; // sentinels

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

**int** count = 0;

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

**if** (arr[i] == arr[j])

count++;

}

// update maxCount if count of

// current element is greater

**if** (count > maxCount) {

maxCount = count;

index = i;

}

}

// if maxCount is greater than n/2

// return the corresponding element

**if** (maxCount > n / 2)

System.***out***.println(arr[index]);

**else**

System.***out***.println("No Majority Element");

}

}

**37)Median of two Sorted Arrays of Different Sizes**

**package** com.ArraysPrograms;

//Median of two Sorted Arrays of Different Sizes

**public** **class** FindMedianArrays {

**public** **static** **void** main(String[] args) {

**int** arr1[]= {1,2,3,4};

**int** arr2[]= {5,6,7,8,9};

**float** fm=*findMedian*(arr1,arr2);

System.***out***.println("Median of two arrays "+fm);

}

**private** **static** **float** findMedian(**int**[] arr1, **int**[] arr2) {

**int** arr3[]=**new** **int**[arr1.length+arr2.length];

**int** i=0, j=0, k=0;

**while**(i<arr1.length&&j<arr2.length)

{

**if**(arr1[i]<arr2[j])

{

arr3[k]=arr1[i];

i++;

k++;

}

**else**

{

arr3[k]=arr2[j];

j++;

k++;

}

}

**while**(i<arr1.length)

{

arr3[k]=arr1[i];

i++;

k++;

}

**while**(j<arr2.length)

{

arr3[k]=arr2[j];

j++;

k++;

}

**if**(arr3.length%2==0)

{

**int** mid=arr3.length/2;

**return**(**float**)(arr3[mid]+arr3[mid-1])/2;

}

**else**

{

**int** mid=arr3.length/2;

**return** arr3[mid];

}

}

}

**38)REMOVE DUPLICATES ELEMENTS FROM A SORTED ARRAY**

**package** com.ArraysPrograms;

//REMOVE DUPLICATES ELEMENTS FROM A SORTED ARRAY

**public** **class** RemoveDuplicatesSortedArray {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5,5};

**int** rd=*removeDuplicates*(arr);

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

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

}

}

**private** **static** **int** removeDuplicates(**int**[] arr) {

**int** rd=0;

**for**(**int** i=1;i<arr.length;i++) {

**if**(arr[rd]!=arr[i]) {

rd++;

arr[rd]=arr[i];

}

}

**return** rd+1;

}

}

**39)FIND NTH BIGGEST ELEMENT FROM AN UNSORTED ARRAY**

**package** com.ArraysPrograms;

//FIND NTH BIGGEST ELEMENT FROM AN UNSORTED ARRAY

**public** **class** NthBiggestInUnsortedArray {

**public** **static** **void** main(String[] args) {

**int** arr[]= {23,45,67,65,43,78,53};

**int** n=5;

**int** big=*nthBiggest*(arr,n);

System.***out***.println(n+"th Biggest element is "+big);

}

**private** **static** **int** nthBiggest(**int**[] arr, **int** n) {

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

**int** count=0;

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

**if**(arr[i]<arr[j]) {

count++;

}

}

**if**(count==n-1)

**return** arr[i];

}

System.***out***.println(n+"is not found");

**return** 0;

}

}

**40)FIND NTH SMALLEST ELEMENT FROM AN UNSORTED ARRAY**

**package** com.ArraysPrograms;

//FIND NTH SMALLEST ELEMENT FROM AN UNSORTED ARRAY

**public** **class** NthSmallestElementInUnsortedArray {

**public** **static** **void** main(String[] args) {

**int** arr[]= {23,45,67,65,43,78,53};

**int** n=4;

**int** big=*nthSmallest*(arr,n);

System.***out***.println(n+"th Smallest element is "+big);

}

**private** **static** **int** nthSmallest(**int**[] arr, **int** n) {

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

**int** count=0;

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

**if**(arr[i]>arr[j]) {

count++;

}

}

**if**(count==n-1)

**return** arr[i];

}

System.***out***.println(n+"is not found");

**return** 0;

}

}

**41)INSERT ONE ARRAY ELEMENTS TO ANOTHER ARRAY IN A GIVEN POSITION**

**package** com.ArraysPrograms;

//INSERT ONE ARRAY ELEMENTS TO ANOTHER ARRAY IN A GIVEN POSITION

**public** **class** InsertArraySpecifiedPosition {

**public** **static** **void** main(String[] args) {

**int** ar[]= {23,45,67,89,87,65,32};

**int** br[]= {33,22,26};

**int** index=3;

*insertArray*(ar,br,index);

}

**private** **static** **int**[] insertArray(**int**[] ar, **int**[] br, **int** index) {

**if**(index<0||index>ar.length) {

System.***out***.println("Index not in the range");

**return** ar;

}

**int** cr[]=**new** **int**[ar.length+br.length];

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

ar[index+i]=br[i];

}

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

**if**(i<index) {

cr[i]=ar[i];

}

**else** {

cr[i+br.length]=ar[i];

}

}

**return** cr;

}

}

**42)FIND FREQUENCY OF ARRAYS ELEMENTS USING CORRESPONDING INDEX POSITION**

**package** com.ArraysPrograms;

//FIND FREQUENCY OF ARRAYS ELEMENTS USING CORRESPONDING INDEX POSITION

**public** **class** FreqElementCorrespondingIndexPosition {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5,1,2};

*printFreq*(arr);

}

**private** **static** **void** printFreq(**int**[] arr) {

**int** max=arr[0];

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

**if**(arr[i]>max) {

max=arr[i];

}

}

**int** ct[]=**new** **int**[max+1];

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

ct[arr[i]]++;

}

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

**if**(ct[i]>0) {

System.***out***.println(i+"-->"+ct[i]);

}

}

}

}

**43)FIND FREQUENCY OF ARRAYS ELEMENTS IN A SORTED ARRAY**

**package** com.ArraysPrograms;

//FIND FREQUENCY OF ARRAYS ELEMENTS IN A SORTED ARRAY

**public** **class** FreqElementsInSortedArray {

**public** **static** **void** main(String[] args) {

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

*findFrequency*(arr);

}

**private** **static** **void** findFrequency(**int**[] arr) {

**int** freq=1;

**int** i=1;

**while**(i<arr.length)

{

**while**(i<arr.length&&arr[i]==arr[i-1])

{

freq++;

i++;

}

System.***out***.println(arr[i-1]+"--->"+freq);

freq=1;

i++;

}

**if**(arr.length==1||arr[i-1]!=arr[i-2])

{

System.***out***.println(arr[i-1]+"--->"+freq);

}

}

}

**44)RETURN ALL THE DUPLICATES ELEMENTS IN A GIVEN ARRAY**

**package** com.ArraysPrograms;

//RETURN ALL THE DUPLICATES ELEMENTS IN A GIVEN ARRAY

**public** **class** ReturnDuplicatesElements {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5,2,4,5};

**int** dp[]=*getDuplicates*(arr);

System.***out***.println("Duplicates elements are ");

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

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

}

System.***out***.println();

}

**private** **static** **int**[] getDuplicates(**int**[] arr) {

**int** dp[]=**new** **int**[arr.length/2];

**int** k=0;

**boolean** rs[]=**new** **boolean**[arr.length];

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

{

**if**(rs[i]==**false**)

{

**int** count=1;

**for**(**int** j=i+1;j<arr.length;j++)

{

**if**(arr[i]==arr[j])

{

count++;

rs[j]=**true**;

}

}

**if**(count>1)

{

dp[k++]=arr[i];

}

}

}

**int** res[]=**new** **int**[k];

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

res[i]=dp[i];

}

**return** res;

}

}

**45)RETURN ALL THE UNIQUE ELEMENTS FROM AN ARRAY**

**package** com.ArraysPrograms;

//RETURN ALL THE UNIQUE ELEMENTS FROM AN ARRAY

**public** **class** ReturnUniqueElements {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5,2,4,5};

**int** un[]=*getUnique*(arr);

System.***out***.println("Unique elements are ");

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

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

}

System.***out***.println();

}

**private** **static** **int**[] getUnique(**int**[] arr) {

**int** un[]=**new** **int**[arr.length/2];

**int** k=0;

**boolean** rs[]=**new** **boolean**[arr.length];

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

{

**if**(rs[i]==**false**)

{

**int** count=1;

**for**(**int** j=i+1;j<arr.length;j++)

{

**if**(arr[i]==arr[j])

{

count++;

rs[j]=**true**;

}

}

**if**(count==1)

{

un[k++]=arr[i];

}

}

}

**int** res[]=**new** **int**[k];

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

res[i]=un[i];

}

**return** res;

}

}

**46)FIND THE MAXIMUM SUBARRAY SUM**

**package** com.ArraysPrograms;

//FIND THE MAXIMUM SUBARRAY SUM(NAIVE APPROACH)

**public** **class** MaximumSubArraySum {

**public** **static** **void** main(String[] args) {

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

**int** ms=*maxSubArraySum*(arr);

System.***out***.println("Maximum subarray sum is "+ms);

}

**private** **static** **int** maxSubArraySum(**int**[] arr) {

**int** maxSum=0;

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

{

**int** sum=0;

**for**(**int** j=i;j<arr.length;j++)

{

sum=sum+arr[j];

**if**(sum>maxSum)

{

maxSum=sum;

}

}

}

**return** maxSum;

}

}

**47)RETURN THE LONGEST PALINEDROMIC NUMBER FROM AN ARRAY**

**package** com.ArraysPrograms;

//RETURN THE LONGEST PALINEDROMIC NUMBER FROM AN ARRAY

**public** **class** FindLongestPalindromeNumber {

**public** **static** **void** main(String[] args) {

**int** arr[]= {121, 2322, 54545, 999990};

**int** lp=*longestPalindrome*(arr);

System.***out***.println("Longest Palindrome Number is "+lp);

}

**private** **static** **int** longestPalindrome(**int**[] arr) {

**int** res=-1;

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

**if**(arr[i]>res&&*isPalindrome*(arr[i]))

res=arr[i];

}

**return** res;

}

**static** **boolean** isPalindrome(**int** n) {

**int** temp=n;

**int** rev=0;

**while**(n!=0)

{

**int** rem=n%10;

rev=rev\*10+rem;

n=n/10;

}

**if**(rev==temp)

**return** **true**;

**else**

**return** **false**;

}

}

**Sum of positive square elements of array**

**package** com.ArraysPrograms;

//Sum of positive square elements of array

**public** **class** ArraySquareSum {

**public** **static** **void** main(String[] args) {

**int** arr[]= {1,2,3,4,5};

**int** sum=*squareSum*(arr);

System.***out***.println("Square sum of Arrays Elements"+sum);

}

**private** **static** **int** squareSum(**int**[] arr) {

**int** sum=0;

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

{

**int** square=arr[i]\*arr[i];

sum=sum+square;

}

**return** sum;

}

}