**BLOCK SWAP**

**import java.util.\*;**

**public class blockswap {**

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

**int n;**

**int k;**

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

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

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

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

**arr[i]=sc.nextInt();**

**}**

**k = sc.nextInt();**

**k = k%n;**

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

**System.arraycopy(arr,0,temp,0,k);**

**System.arraycopy(arr,k,arr,0,n-k);**

**System.arraycopy(temp,0,arr,n-k,k);**

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

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

**}**

**}**

**}**

**BOOTH**

**import java.util.Scanner;**

**public class Booth {**

**public int multiply(int n1, int n2) {**

**int m = n1;**

**int r = n2;**

**int A = n1;**

**int S = -n1;**

**int P = 0;**

**int count = Integer.SIZE;**

**System.out.print(count);**

**while (count > 0) {**

**if ((r & 1) == 1) {**

**P += A;**

**S += m;**

**}**

**A <<= 1;**

**S <<= 1;**

**count--;**

**r >>= 1;**

**}**

**return P;**

**}**

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

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

**Booth b = new Booth();**

**System.out.println("Enter two integer numbers -");**

**int n1 = scan.nextInt();**

**int n2 = scan.nextInt();**

**int result = b.multiply(n1, n2);**

**System.out.println("\n\nResult : " + n1 + " \* " + n2 + " = " + result);**

**}**

**}**

**GCD**

**import java.util.Scanner;**

**public class GCD {**

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

**int n1, n2;**

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

**n1 = sc.nextInt();**

**n2 = sc.nextInt();**

**if (n1 < n2) {**

**int temp = n1;**

**n1 = n2;**

**n2 = temp;**

**}**

**int result = gcd(n1,n2);**

**System.out.println("The GCD of " + n1 + " and " + n2 + " is: " + result);**

**}**

**public static int gcd(int n1, int n2) {**

**if(n1 == 0){**

**return  n2;**

**}**

**return gcd(n2%n1,n1);**

**}**

**}**

**LONGEST 1**

**import java.util.Scanner;**

**public class Longest1 {**

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

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

**String binaryString = sc.next(); // Read input as a string**

**int count = 0, maxCount = 0;**

**for (char i : binaryString.toCharArray()) {**

**if (i == '1') {**

**count++;**

**if (count > maxCount) {**

**maxCount = count;**

**}**

**} else {**

**count = 0;**

**}**

**}**

**System.out.println("The longest sequence of 1's is: " + maxCount);**

**}**

**}**

**BINARYPALINDROME**

**import java.util.Scanner;**

**public class Binarypalindrome {**

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

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

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

**String binaryString = Integer.toBinaryString(x);**

**String reversedString = new StringBuilder(binaryString).reverse().toString();**

**if (binaryString.equals(reversedString)) {**

**System.out.println(x + " has a binary palindrome representation.");**

**} else {**

**System.out.println(x + " does not have a binary palindrome representation.");**

**}**

**}**

**}**

LEXI

public class Main

{

public static void main(String[] args) {

String s = "malayalam";

int len = s.length();

int max = 26;

int freq[] = new int[max];

// to find the frequency of each characters

for(int i=0;i<len;i++)

freq[s.charAt(i)-'a']++;

if(CanMakePalindrome(freq,max,len) == false)

{

System.out.println("No palindromic string");

System.exit(0);

}

//code to reduce odd\_count character

String odd\_str="";

for(int i=0;i<max;i++)

{

if(freq[i]%2!=0)

{

freq[i]--;

odd\_str =odd\_str + (char)(i+'a');

}

}

//code to create palindromic string

String f="",r="";

for(int i=0;i<max;i++)

{

if(freq[i]!=0)

{

String temp = "";

char ch = (char) (i+'a');

for(int j=1;j<=freq[i]/2;j++)

{

temp = temp + ch;

}

f = f+temp;

r = temp+r;

}

}

System.out.println(f+odd\_str+r);

}

public static boolean CanMakePalindrome(int freq[],int max,int len)

{

// code to check whether it is possible to palindromic string with the given string

int odd\_cnt=0;

for(int i=0;i<max;i++)

{

if(freq[i]%2!=0)

odd\_cnt++;

}

if(len%2 == 0)

{

if(odd\_cnt>0)

{

return false;

}

else

{

return true;

}

}

else

{

if(odd\_cnt!=1)

return false;

}

return true;

}

}

**karathsubbha**

**public class KaratsubaMultiplication {**

**public static long karatsubaMultiply(long x, long y) {**

**// Base case for recursion**

**if (x < 10 || y < 10) {**

**return x \* y;**

**}**

**// Calculate the size of the numbers**

**int n = Math.max(Long.toString(x).length(), Long.toString(y).length());**

**int half = (n + 1) / 2;**

**// Split the digit sequences about the middle**

**long a = x / (long) Math.pow(10, half);**

**long b = x % (long) Math.pow(10, half);**

**long c = y / (long) Math.pow(10, half);**

**long d = y % (long) Math.pow(10, half);**

**// Perform the three multiplications**

**long ac = karatsubaMultiply(a, c);**

**long bd = karatsubaMultiply(b, d);**

**long adbc = karatsubaMultiply(a + b, c + d) - ac - bd;**

**// Combine the results**

**return (long) (ac \* Math.pow(10, 2 \* half) + adbc \* Math.pow(10, half) + bd);**

**}**

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

**long x = 12345678;**

**long y = 87654321;**

**long result = karatsubaMultiply(x, y);**

**System.out.println("The product of " + x + " and " + y + " is: " + result);**

**}**

**}**

**HOURGLASS**

**import java.util.Scanner;**

**public class Hourglass {**

**// Function to find the maximum sum of an hourglass in a given matrix**

**public static int findMaxSum(int [][]mat, int R, int C) {**

**// Initialize max\_sum to the smallest possible value and sum to 0**

**int max\_sum = 0;**

**int sum;**

**// If the matrix is smaller than 3x3, it's not possible to form an hourglass**

**if (R < 3 || C < 3) {**

**System.out.println("Not possible");**

**System.exit(0);**

**}**

**// Iterate over all possible positions of the top-left element of the hourglass**

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

**for (int j = 0; j < C - 2; j++) {**

**// Calculate the sum of the current hourglass**

**sum = (mat[i][j] + mat[i][j + 1] + mat[i][j + 2]) // Top row**

**+ (mat[i + 1][j + 1])                     // Middle element**

**+ (mat[i + 2][j] + mat[i + 2][j + 1] + mat[i + 2][j + 2]); // Bottom row**

**// Update max\_sum if the current hourglass sum is greater**

**max\_sum = Math.max(max\_sum, sum);**

**}**

**}**

**// Return the maximum hourglass sum found**

**return max\_sum;**

**}**

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

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

**// Reading the dimensions of the matrix**

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

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

**// Initializing the matrix**

**int[][] mat = new int[R][C];**

**// Reading the elements of the matrix**

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

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

**mat[i][j] = sc.nextInt();**

**}**

**}**

**// Finding and printing the maximum hourglass sum**

**System.out.println("Maximum hourglass sum: " + findMaxSum(mat, R, C));**

**}**

**}**

**RIGHT LEADER**

**import java.util.\*;**

**public class RightLeader {**

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

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

**System.out.print(rmax+" ");**

**for(int i=n-2; i>=0; i--){**

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

**rmax=arr[i];**

**System.out.print(rmax+" ");**

**}**

**}**

**}**

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

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

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

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

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

**arr[i] = scan.nextInt();**

**}**

**Leader(arr,n);**

**}**

**}**

**MAX SUB ARR**

**import java.util.\*;**

**public class maxprodsubarr {**

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

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

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

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

**for(int  i=0; i<n; i++){**

**arr[i] = scan.nextInt();**

**}**

**int prefix =1;**

**int suffix =1;**

**int ans =1;**

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

**if(prefix==0){**

**prefix =1;**

**}**

**if(suffix==0){**

**suffix=1;**

**}**

**prefix = prefix\*arr[i];**

**suffix = suffix\*arr[n-i-1];**

**ans = Math.max(ans, Math.max(prefix, suffix));**

**}**

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

**}**

**}**

**NIBBLE SWAP**

**import java.util.\*;**

**public class nibbleswap {**

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

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

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

**int swapnum = ((n & 0x0F) <<4 | (n & 0xF0)>>4);**

**System.out.println("Number before swap: "+n);**

**System.out.println("Number after swap: "+swapnum);**

**}**

**}**

**MAJ ELEM**

**import java.util.\*;**

**public class MajElem {**

**public static int Major(int[] arr){**

**int m=-1;**

**int i=0;**

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

**if(i==0){**

**m = arr[j];**

**System.out.println("m "+m);**

**i=1;**

**System.out.println("i "+i);**

**}**

**else if(m==arr[j]){**

**System.out.println("m "+m);**

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

**System.out.println("i "+i);**

**i++;**

**System.out.println("i "+i);**

**}**

**else{**

**i--;**

**System.out.println("i "+i);**

**}**

**}**

**return m;**

**}**

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

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

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

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

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

**arr[i] = scan.nextInt();**

**}**

**System.out.println("MAJOR ELEMENT"+Major(arr));**

**}**

**}**